A COMPUTERIZED SYSTEM FOR OBTAINING A DECISION TABLE DESCRIPTION OF A PROGRAM

VLADIMIR HOLLMANN

A Thesis

in

The Faculty

of

Engineering

Presented in Partial Fulfillment of the Requirement
for the degree of Master of Engineering at
Concordia University
Montréal, Québec, Canada

September, 1975

ABSTRACT

VLADIMIR HOLLMANN

A COMPUTERIZED SYSTEM FOR OBTAINING A DECISION TABLE DESCRIPTION OF A PROGRAM

A system has been implemented which has the capability of translating programs written in the FORTRAN language into decision tables. Listings of original source code, all non-executable statements, decision tables, and table cross-reference are also produced.

The system has been written in a modular fashion to conserve core memory and facilitate the addition of extra features. It is driven by control cards which specify the functions to be performed at a given execution.

Decision tables are represented by records in two files referred to as an index file and an information file. The records in the index file contain pointers giving the key of the first record of each type for that table in the information file. Records in both files are arranged in a doubly-linked list configuration.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

ACKNOWLEDGEMENT

I wish to thank Professor Wojciech M. Jaworski, Associate Professor of the Department of Computer Science, Concordia University for his supervision, remarks, and advice on this project.

INTRODUCTION

a) History of the Project

In March of 1971 the author began investigating topics for a paper in partial fulfillment of the requirements for the degree of Master of Engineering at Concordia University. The subject of decision tables and, in particular, the translation of source programs into decision tables was of great interest as it was felt that a process performing this function would be a useful tool both for program development and for documentation. Such a tool did not exist at the time. The objective of developing such a process was finalized in November 1971 and work on the project commenced.

As a system of this nature would involve character manipulation it was decided to write it in the PL/I language. The author had access to an IBM 360/65 com uter at that time and did not foresee any difficulties with this choice.

In the summer of 1972 the author changed his employer and no longer had access to I M hardware. He was forced to discard all programming done to date and begin again as the machine now available, a CDC 6600, did not support the PL/I language.

By the winter of 1972 work had once again reached the stage earlier attained, however, the author's workload in his employment as a consultant became so heavy that no

work could be done on the project for a period of one year.

The project was resumed again in the spring of 1974 and was finally completed in the spring of 1975.

b) Decision Tables and Flowcharts

A limited entry decision table defines a set of rules such that for a given set of conditions a corresponding set of actions is carried out. Similarly, Kavanaugh [7] defines a decision table as ". . . a precise statement of both the logical and quantitative relationships supporting a particular elementary decision.". In some cases, all actions in a decision table are to be carried out unconditionally. A table of this type, consisting of actions only, will be referred to as an "action table".

A flowchart consists of a set of standard symbols containing a description of operations being performed which are physically interconnected with directional lines thereby providing a visual indication of the logic involved. As with decision tables, flowcharts may be produced to describe logic at as detailed or general a level as required.

There are several systems, such as Autoflow of Applied Data Research Inc. and Flowtrace [11] which translate program source code to flowcharts. Some of these are discussed by Chapin [3]. The primary purpose of such systems is to provide program documentation.

A major disadvantage of flowcharts is their sequential nature. This can easily lead to a situation in which the

connections between blocks are so disorganized as to render the flowchart useless from the point of view of being an instrument of clarification of the source program. Decision tables, on the other hand, do not impose any requirement for showing a physical connection between logical blocks. The physical order of presentation of the tables is independent of the logic and so is entirely arbitrary. Thus a set of decision tables describing a given program may be written with greater clarity than is possible with flowcharts.

A second important advantage of decision tables versus flowcharts is their suitability for clearly and concisely describing complex logic situations. This means that a decision table analysis of a program will quickly point up flaws in the logic.

Other benefits of decision tables may be found in Kavanaugh [6] , and Schmidt and Kavanaugh [10] .

c) Source Code Translation

The purpose of a system for the translation of programs to decision tables is twofold. Not only will it provide clear and concise program documentation but it is also an aid to program development and analysis.

Although much work has been done on the process of translating decision tables to source code by people such as Muthukrishnan and Rajaraman [9] , King and Johnson [8] , and Ganapathy and Rajaraman [5] , to date there has been little literature available dealing with the reverse process, namely that of translation of program source code to decision

tables. Cavouras [2] has investigated the subject, however, his approach seems to have been more specialized than the system described in this paper. He appears to have developed a system which operates on the source program directly to produce decision tables. Thus it would seem to lack flexibility in terms of possible future enhancements providing increased capabilities both in terms of input handling as well as the range of functions performed. The system described herein was developed with a view to flexibility to permit the easy addition of such future enhancements as increased language handling capability and an interactive mode of operation.

As the objective here was to produce a working system, this paper is oriented toward describing the practical aspects of this system and so does not cover the more theoretical aspects of structured programming, transformations, table connectivity, etc.

I. THE SYSTEM

a) Philosophy

As the proposed system was to be a practical, flexible, multi-purpose system, two decisions were initially taken. The first was that the entire system would be written in the COBOL language. This was done due to the portability of COBOL systems between machines of different manufacturers. A secondary reason was that COBOL is better suited to character manipulations than is FORTRAN, the other language under consideration.

The second decision was that the entire system would be written in a modular fashion. This approach has several advantages:

1. The front-end language translator is the only language-dependent portion of the system. In order to process programs written in other languages, it is necessary only to supply an appropriate translator module. The rest of the system remains unchanged.
2. The entire system is driven by a monitor program. This approach permits a great degree of flexibility and control over the functions to be performed.
3. Additional features are readily integrated into the system through the addition of new modules.
4. The individual portions of the system will be required

at different times or perhaps not at all. This permits us to overlay many of the modules and so conserve core memory.

5. A modular system is relatively simple to develop and maintain due to the independence of the individual modules and localization of functions within them.

Keeping in mind the level of flexibility desired in this system, the following operating concept governed the design. The original source program is broken down into the simple configuration of limited entry decision tables having at most one condition stub. It is then possible to combine these tables or otherwise operate on them in any way desired. with a minimum of difficulty.

In order to accomplish this, every conditional statement in the source program must be represented in the following form:

<name>
IF <condition>
THEN <action>
ELSE <action>

Names are generated by the system every time a condition record is written as well as every time a label is encountered in the source program. Each name defines the start of a new decision table. Transfers between tables are accomplished by means of the GO TO <name> construct. Any comments encountered in the original source code are transformed into records of whatever type immediately preceded them.

This procedure yields the desired elementary decision tables.

With the emphasis in programming techniques these days being on structured programming which, as discussed by Dijkstra [4] , can reduce the problem of program correctness, there was a concern that programs originally written in a structured fashion should remain so after processing by the system. It was felt that the system as implemented satisfied this criterion as no logic changes take place and the effect of combining some tables with others is one which leads towards rather than away from a structured format. This may not, however, be the case if some future enhancements, such as user control of table manipulation discussed elsewhere in this paper, are implemented.

b) Operation

The current system as implemented consists of six modules as shown in fig. 1. These are the following:

1. DETBL00 is the monitor module. It performs all the control functions for the system.
2. DET3L40 is the FORTRAN language translator. This module reads the source program and generates a language independent work file.
3. DETBL10 works in conjunction with the language translator. It reads the work file produced and generates the decision table files.
4. DET3L20 operates on the decision table files in such a way as to merge two decision tables into one provided they satisfy a given set of requirements.
5. DET L30 is the module which generates the decision table

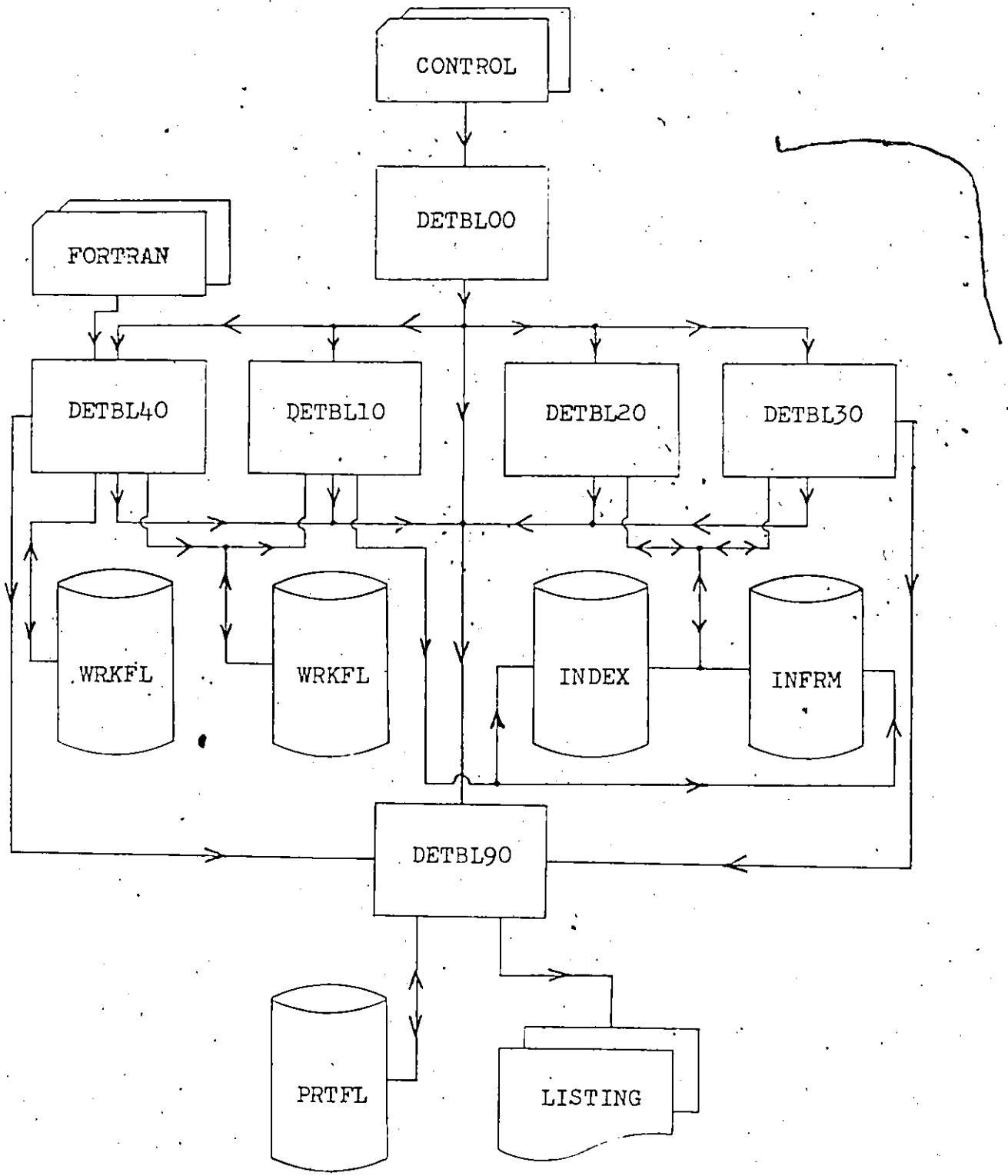


Fig. 1. The DETABLS system

listing and cross-reference map.

6. DETBL90 is a print utility module. It accumulates print-image records generated by the other modules and prints all requested listings at the end of the run.

The modules DETBL10, 20, 30, 40 and a portion of DETBL90 are independent. In order to conserve core memory the system is set up in such a way that these modules overlay each other as they are required.

As currently implemented, this system can perform the following functions:

1. Generate elementary decision tables from FORTRAN source code.
2. Combine tables where specific circumstances, described later in this paper, prevail.
3. Produce the following four listings:
   a. Source listing - This is a listing of the original FORTRAN source code being translated.
   b. Non-executable statements - Any non-executable statements encountered in the source program, with the exception of comments, are not included in the decision tables generated. These statements are listed separately to provide a record of all statements omitted from the decision table files.
   c. Decision tables - A listing of the resulting decision tables.
   d. Cross-reference map - This listing shows for each table the names of all tables to which it refers as

well as those tables which refer to it.

The system is driven by a control file of parameter cards. These instruct it as to the functions to be performed and the outputs desired.

The source code is read and a work file generated. This file is processed in order to resolve label references and a second work file is produced. This second file corresponds on a one to one basis to the decision tables desired, however, further processing takes place in order to obtain the final decision table files.

Once the final files have been produced, DETBL20 may be called upon to merge certain tables if conditions permit. This can be a multi-pass process. The number of passes is under the control of the user.

If desired, the table listing module, DETBL30, may be called upon to produce a table listing and cross-reference map at any stage after the tables have been created.

When all requested operations have been performed, the print utility module is invoked and the listings printed.

c) File Structure

Having decided on a system configuration, we are faced with the problem of representing a decision table in a format suitable for computer processing. The criteria governing a file design were the following:

1. Each table must be accessed randomly.
2. Any desired information pertaining to a table must be available randomly.

3. Decision tables may be processed in forward or reverse sequence.
4. Tables or portions of tables may be added, deleted, altered, or moved to other tables.

For this application, separate files would be required for each type of information associated with a given table. This approach would, however, lead to a multitude of files and cause severe problems during processing. The desired effect can be obtained by using random access files. By chaining together records which are logically connected and supplying a pointer to the first one we obtain the effect of many logical files within a single physical file.

The file structure adopted is similar to an inverted list structure. It consists of two physically separate files; an index file and an information file. The index file contains records giving pointers to the first record of each type within the information file for the associated decision table. Fields containing counts of the number of conditions, actions, and rules also appear on the pointer record.

Access to the file is obtained through the header record which has the key 'FIRST'. The file is terminated by a trailer record accessed with the key 'AVAIL'. Records, with the exception of the header and trailer, have the format shown in fig. 2. Each record contains a pointer to the next as well as previous records. The header record contains blanks in the previous record pointer field to

A	B	C	D	E	F	G	H	I	J	K	L

LEGEND

A - Table name

B - Next table name in list

C - Previous table name in list

D - Pointer to name information record

E - Pointer to 'REFERS TO' information record

F - Pointer to 'REFERENCED BY' information record

G - Pointer to condition information record

H - Pointer to action information record

I - Total number of line items in table

J - Total number of actions in table

K - Total number of conditions in table

L - Total number of rules in table

Fig. 2. Pointer record

indicate that there is no previous record and the trailer contains '99999' in the next record pointer field to indicate the end of the list. The structure is that of a doubly-linked list as shown in fig. 3.

The trailer record does not contain pointers to information file entries but rather contains three fields; a count of the total number of decision tables, the index of the first unused record in the information file, and the index of the last unused record.

There are five types of records in the information file as shown in fig. 4. A minimum of one record of each of four types is required to define an action table and at least one record of each of the five types defines a decision table. The pointer record for the table contains the index of the first record of each type associated with the table. Subsequent records of each type are linked to the previous ones in a doubly-linked list configuration. A pool of unused records, also arranged as a doubly-linked list, is accessed via the 'AVAIL' pointer record.

The first type of record used to define a table is the name record. It contains the generated table name followed by the last label encountered within the source program before generation of the table. This feature permits the easy relation of the decision table to the original source code.

Another type of record gives the number of decision tables to which the associated table refers as well as the

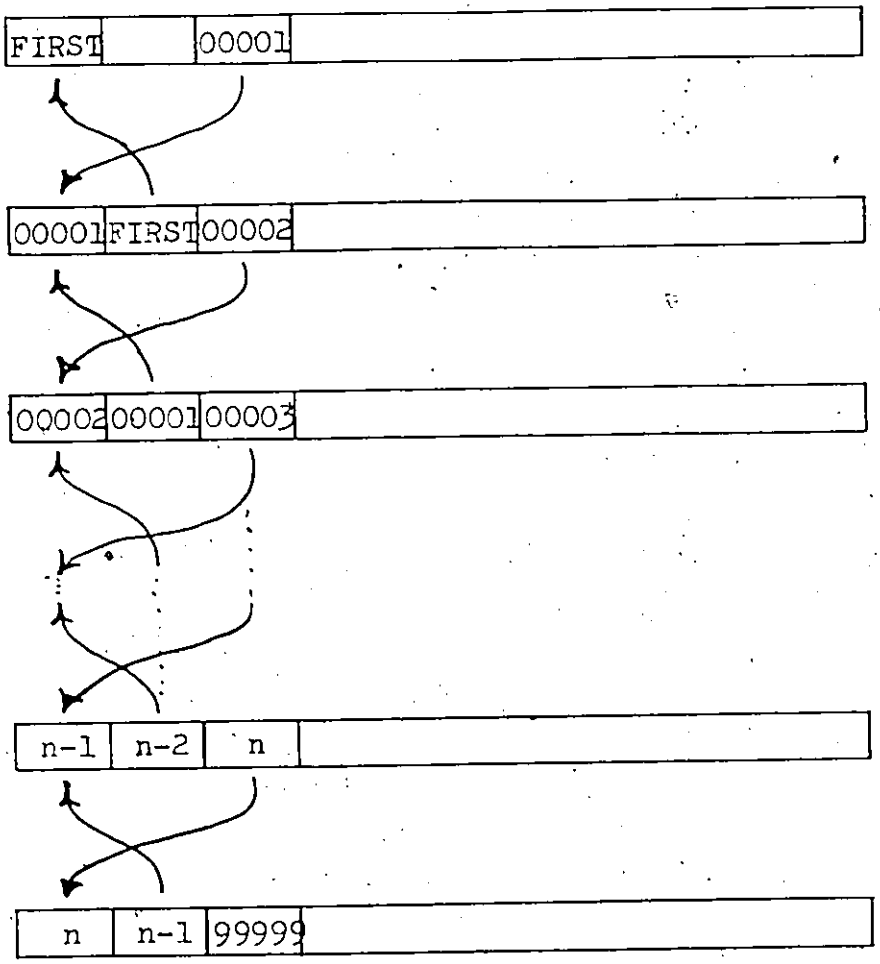
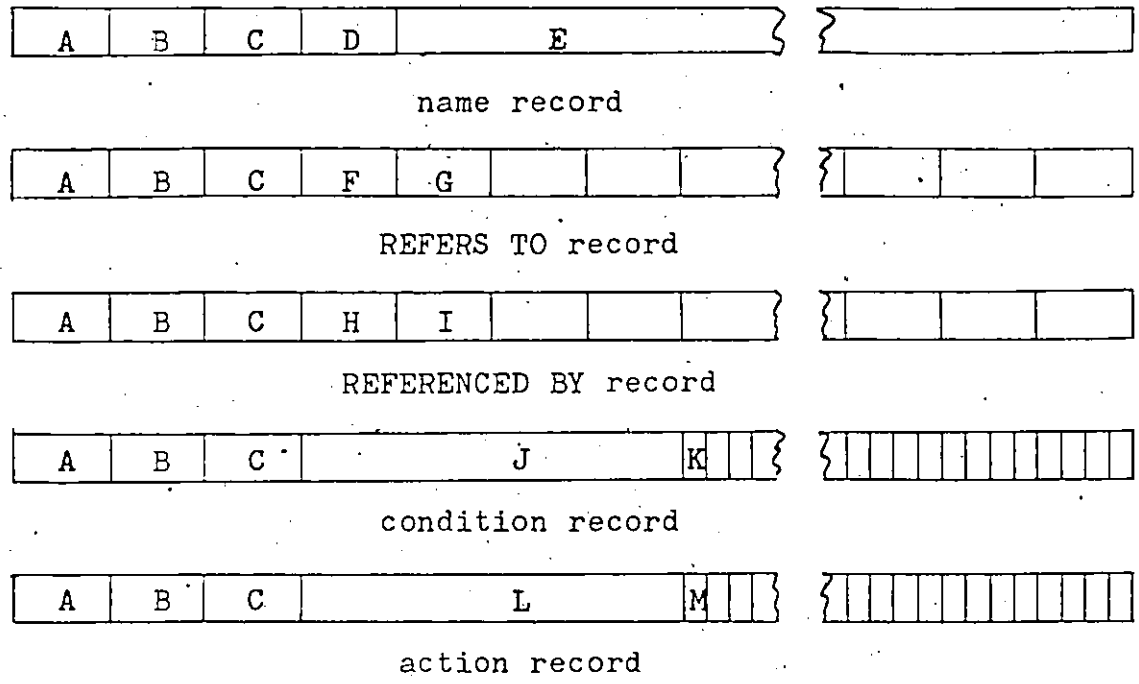


Fig. 3. Doubly-linked list



LEGEND

A - Record key

B - Key of previous record

C - Key of next record

D - Table name

E - Last label encountered in source code

F - Number of tables to which this one refers

G - Names of tables to which this one refers (max. 20/recd.)

H - Number of tables referring to this one

I - Names of tables referring to this one (max. 20/recd.)

J - Condition stub

K - Condition entry (maximum 40 rules)

L - Action stub

M - Action entry (maximum 40 rules)

Fig. 4. Information records

names of these tables. A third type of record, similar to this one, lists the number as well as the names of tables which refer to the associated decision table. Together these two types of records define the relationships between tables and facilitate processing.

Finally, tables consist of condition and action records each of which contain both the stub and entry portions of the decision table format. They are stored in print image format to facilitate processing. As the index and information files generated in this way are relatively small, it was felt that the simplified processing this arrangement makes possible outweighs the slightly larger amount of disk space required by this method.

The records are designed to accomodate a maximum of forty rules. This figure was chosen for two reasons:

1. It is within the limit required to fit the stub and entry portions of conditions and actions on one line of print.
2. The more rules a decision table contains, the more confusing it becomes thus undermining the main feature of a decision table -- clarity of presentation.

Normally a decision table will be defined by a pointer record and one or more of each of the five types of information records described above. This is shown in fig. 5. In the case of an action table, however, no condition records exist. This fact is indicated by the presence of a zero in the condition index field of the pointer record. Thus each table

The table

00100 - TABLE 1		
I.EQ.J	Y	N
GO TO 00200	X	
GO TO 00300		X

would be represented by the following file entries:

00100	00200	FIRST	A	B	C	D	E	3	2	1	2	Pointer
A	00000	00000	00100 - TABLE 1									Name
B	00000	00000	2	00200	00300							Refers to
C	00000	00000	0									Referenced by
D	00000	00000	I.EQ.J							Y	N	Condition
E	00000	F	GO TO 00200							X		Actions
F	E	00000	GO TO 00300								X	

Fig. 5. Correspondence between a decision table and file records

will consist of four or five short doubly-linked lists of information records where the first record of each list is given by the pointer record for that table. This arrangement allows very easy access to any data associated with the table and permits great flexibility in processing.

II. THE MONITOR - DETBLOO

The DETABLS system was designed as a modular system. As such it requires a control module to drive it. This is the function of the monitor.

On initial entry, this module issues a call to the print utility module to initialize it and, on return, proceeds with processing of the control file one card at a time. As each control card is read it is verified. Whenever an error is detected, an attempt is made at recovery. If this is not possible, the card is rejected and the run may be terminated in some cases. Once the control card has been verified, the appropriate module is called.

On return from any module invoked, the successful completion of that module's functions is tested by means of a return code. If any module does not complete its functions successfully, processing is terminated.

Once all control cards have been processed or if premature run termination is forced, the print utility is again invoked to print all the required listings.

III. THE TRANSLATOR - DETBL40

a) Purpose

The DETABLS system was designed in such a way that it can accomodate programs written in various languages. This is accomplished by localizing the language-dependent portion of the system in a translator subroutine which reads the source code and translates it to a standard language-independent format. The resulting file is then used by the system to generate the final decision table files.

b) The File

The language-independent work file generated by the translator consists of seven different types of records identified by a type code in the first character position. These are as follows:

A - Unconditional action - Any statements encountered in the source code which do not contain an implicit or explicit condition test and immediately follow an existing or generated label cause the creation of this type of record.

C - Condition - A condition record is generated for every condition test encountered in the source program.

R - Result - These records consist of all statements to be executed if the condition test in the associated

condition record is satisfied.

E - Else - These records consist of all statements to be executed if the condition test in the associated condition record is not satisfied.

N - Name - A name record is written for every existing or system-generated label encountered.

- - Continuation - These records are generated whenever the associated first record cannot contain the complete information.

* - Comment - Any comments encountered in the original source code cause the generation of this type of record.

Within the work file associated records appear in groups which bear a direct relationship to a decision table. There are two types of groups; those which contain conditions and those which do not.

The first record of each group is the 'N' or name record. This is followed by either an 'A' (action) or 'C' (condition) record. If it is the 'A' type record, no other record types except comments or continuations may appear in the group. In the case of a 'C' type record, this will be followed by at least one 'R' and one 'E' record. As with the group of action records comments or continuations may appear anywhere.

In both types of group, the last record of each of the 'A', 'R' and 'E' types must contain either the 'END' or the 'GO TO' construct.

c) Operation

In order to obtain the previously described file structure from a source program it is first necessary to transform all FORTRAN statements involving a condition test into the general form of

<name>
IF <condition>
THEN <action>
ELSE <action>

For the purposes of this translator, FORTRAN statements may be divided into the following four distinct categories:

1. Comments - Comment cards
2. Non-executable - Purely passive statements which do not cause the generation of any machine language instructions.
3. Conditional - Statements which either implicitly or explicitly cause a condition to be tested.
4. Unconditional - Statements which do not fall into the previous two categories automatically belong to this one.

A complete list of statements comprising each category is available in appendix 'A'.

As the translator reads the source code, it calls the print utility module to print each card read if requested. It then checks for any comment cards which may be present. When encountered, these require no further processing and so are stored in a table until the current statement has been read and processed. Any comments accumulated in the table are then written out.

Other cards are read into a table until the complete statement has been stored. Provision has been made for a statement including up to nineteen continuation cards. Once the entire statement has been stored, analysis may proceed.

The first eight non-blank characters are matched against keywords in order to identify the statement category. A match on eight characters is sufficient as FORTRAN variable names are limited to seven characters in length. Any match obtained with eight characters must therefore be the desired keyword. The majority of non-executable statements can be identified in this way. Others are treated as special cases and analyses are made to determine whether they are non-executable or other types of statements. Any non-executable statements encountered are listed, if desired, and dropped from the file.

Once a given statement has been found to be executable, a test is made to determine whether it is in the conditional or unconditional category. There are only three types of conditional statements to be considered. These are the 'DO' statement, computed and assigned 'GO TO', and the various forms of the 'IF' statement listed in appendix 'C'. Any statement not found to be one of these is automatically in the unconditional category.

d) IF Statement

The 'IF' statement occurs in two basic forms. One of these includes the action to be executed if the condition is satisfied as part of the statement itself. This type

of statement is handled as follows:

1. If a label does not exist, one is generated. A name record is then written to the work file.
2. The condition is extracted and a condition record produced.
3. A result record is generated containing the resulting action.
4. The next name to appear in the file is determined by examining the next executable statement in the source program for a label. The name obtained is inserted into a 'GO TO' construct and written out as an else record.
5. If the statement examined above is not labelled, a name record with the generated name is created.

The second form of the 'IF' statement consists of an expression to be evaluated and a list of transfer addresses. Control is passed to one of these addresses depending on the value assumed by the expression.

Statements of this type are handled in such a way that, if the expression may assume more than two values, additional condition test are generated so that each expression will only assume the values 'TRUE' or 'FALSE'. Each condition test generated is given a label (transfer address). An example showing the conversion of an arithmetic 'IF' statement is shown in fig. 6.

Other types of 'IF' statement are similarly exploded, if necessary, into separate condition tests and the appropriate transfer addresses are generated.

The FORTRAN statement

```
    5    IF(X) 10,20,30
```

becomes the following:

```
001005 IF(X.LT.0)
           THEN GO TO 10
           ELSE GO TO 001015
001015 IF(X.EQ.0)
           THEN GO TO 20
           ELSE GO TO 30
```

Fig. 6. Explosion of arithmetic 'IF' statement

e) Computed and Assigned GO TO

The computed and assigned GO TO statements are handled similarly to the arithmetic IF statement described previously. Labels are extracted from the list and records generated testing the associated variable for the values 1, 2, etc. in the case of the computed GO TO and for the values of the labels in the case of the assigned GO TO. If the condition test is satisfied, a branch is taken to the appropriate label. If it is not, the next value is tested. Labels are generated at the start of each condition test.

Whenever none of the conditions are satisfied, a branch is taken to the next executable source statement in the case of the computed GO TO. An assigned GO TO statement will cause the generation of the END construct in this case.

Examples of the explosion of computed and assigned

'GO TO' statements are given in figs. 7 and 8 respectively.

The FORTRAN statements

```
      10    GO TO (20,30)I
            I=I+1
```

becomes the following:

```
      0010010 IF(I.EQ.1)
              THEN GO TO 20
              ELSE GO TO 0010110
      0010110 IF(I.EQ.2)
              THEN GO TO 30
              ELSE GO TO 0010210
      0010210 I=I+1
```

Fig. 7. Explosion of computed GO TO statement

The FORTRAN statement

```
      10    GO TO M,(20,30)
```

becomes the following:

```
      0010010 IF(M.EQ.20)
              THEN GO TO 20
              ELSE GO TO 0010110
      0010110 IF(M.EQ.30)
              THEN GO TO 30
              ELSE END
```

Fig. 8. Explosion of assigned GO TO statement

f) DO Statement

This statement does not bear any similarity to the 'IF' or 'GO TO' statements and is treated in an entirely different manner.

Basically, the 'DO' statement can be considered as generating three functions: initialization, incrementing, and testing. The initialization is performed once and does not really constitute a part of the 'DO loop', however, the test and increment functions are performed as the last functions within the loop on each pass.

In order to translate the 'DO' statement into the required form, we must first generate an initializing action record before the first statement within the range of the 'DO'. This record is immediately followed by a name record.

At the end of the range of the 'DO', immediately following the last statement, we insert a test to determine if the variable has reached its maximum value. Branches are generated so that if it has not, the variable is incremented by the required amount and control returned to the beginning of the loop. If it has reached the maximum value, control will pass to the next statement. A label will be generated if one does not already exist. Fig. 9 gives an example of this process.

Nested 'DO' loops are handled by storing the condition test, increment information, and branch labels in a push-down stack every time a 'DO' statement is encountered. Then, as the end of the range of each 'DO' is reached, the information

The FORTRAN statements

```
5       P=Q
        DO 10 I=1,5,2
        J=J+1
10      CONTINUE
        I=0
```

become the following:

```
001005 P=Q
       I=1
001015 J=J+1
0020010 CONTINUE
0020110 IF(I.GE.5)
          THEN GO TO 0020210
          ELSE I=I+2
          ELSE GO TO 001015
0020210 I=0
```

Fig. 9. Explosion of DO statement

is retrieved and inserted in the appropriate place in the file.

g) Other Statements

All executable statements other than the three special cases previously discussed are written out to the work file as 'A' type records. A name record precedes the first action record in each group. Generation of the group is terminated whenever a conditional or labelled statement is encountered in the source code. This causes the creation of a new name record starting a new group.

h) Labels

Labels encountered in the source code cause 'N' type or name records to be written to the work file. Names are obtained by prefixing the coded label with a five-digit table name. This is incremented to the next highest multiple of one hundred each time a label is encountered in the source code.

On occasion it is necessary to generate extra names. This is accomplished by adding one to the five-digit prefix of the name as last written.'

This transformation of labels means that the references within the work file to labels or table names are no longer accurate due to the prefixing operation. It then becomes necessary to process the work file generated and supply the correct system-generated names to the 'GO TO' constructs which refer to the original statement labels.

As each label is encountered in the source code and a

corresponding table name generated, both values are stored in a cross-reference table. It is then possible, using this table, to process the work file and replace all references to original labels with the new names. Any 'R', 'E', or 'A' records containing the 'CONTINUE' statement are deleted at this time as this statement serves no useful function.

At this point, the interpreter has completed its task and control is returned to the monitor. Any further operations on the work file are entirely language independent.

IV. THE FORMATTER - DETBL10

a) Function

The function of the formatter module is to transform the file produced by the translator into the final decision table files -- the index file and the information file. Although this function could have been incorporated in the translator module, this would mean that any additional translators written would duplicate this function. As this procedure is not dependent on the original source language it was decided to isolate it in a separate module of its own. This leads to the unnecessary creation of a second work file by the translator which must then be read by the formatter, however, as this is a relatively small file, the additional overhead introduced may be considered negligible.

b) Operation

The formatter reads the work file and initiates a new table every time a name record (type 'N') is encountered. This causes the pointer record as well as any condition or action records outstanding from the previous table to be written. A new pointer record is initialized and the name record written.

Any type 'C' records encountered cause the generation of condition records in the information file with provision

for two rules corresponding to the 'Y' and 'N' values of the condition entry. Records of types 'R', 'E', and 'A' cause action records to be created in the information file. Type 'R' records cause a flag to be placed in the rule position corresponding to the 'Y' in the condition record while type 'E' records flag the rule position corresponding to 'N'. Records of the type 'A' flag the first rule position in all cases as they are only encountered when generating action tables.

Comments and continuations (record types ' ' and '-') cause the creation of records identical in type to those which immediately preceded them.

A count is kept of the number of conditions, actions, and rules in each table. These counts are stored in the pointer record.

Associated with each table is a list of those tables to which it refers and tables which refer to it. As each record other than a condition record or name record is read, it is examined to determine if it contains the 'GO TO' construct. If it does, the corresponding entry is generated in the 'refers to' list.

The lists of references are in the form of records capable of accomodating 20 table names each. If a list contains more than 20 names, additional records are generated. Each record contains a count of the total number of tables in the list.

In order to generate the list of tables referring

to the one under consideration, the 'refers to' records of each table are read in turn. The names of the tables referenced are retrieved and their associated 'referenced by' records accessed. The name of the referring table is added to the list of each and the records written back out.

Future versions of the DETABLS system may permit the addition of tables. To facilitate this, once all required information file records have been written, a chain of unused records is written out to the full limit permitted. These records form a pool of unused space within the file which may at a later date contain valid information. The first and last records of this chain are pointed to by the 'AVAIL' pointer record. This pointer also contains a count of the total number of decision tables comprising the program.

V. THE MERGE MODULE - DETBL20

a) Function

The decision tables produced by the formatter module contain at most one condition test which is either satisfied or not. These tables contain only one or two rules. In many cases tables have been generated which could be combined with others to improve clarity.

The function of this module is to combine tables where possible so as to improve clarity.

b) Merge Criteria

Processing begins with the index record which has the key 'FIRST'. Tables are then processed in the sequence in which they are encountered in the chain.

All tables referenced by the one being processed are examined in order to determine if it is possible to merge the two. If the merge criteria are satisfied, the tables are combined.

The action of combining two tables may make other tables eligible for merging which previously did not meet the requirements. Thus it may be necessary to make several passes through the index file before all possible merges have been made. The maximum number of passes to be attempted is specified by the 'MERGE' control card. If, however,

the entire index file is processed once without a merge of tables taking place, no further passes are made.

The following are the criteria which must be satisfied in order for a merge of two tables to take place:

1. The table referred to must not be referenced by any other tables. Should this rule not be imposed, it would be possible to merge the two tables, however, in order to eliminate the table referenced it would have to be replicated and merged individually with all tables referring to it. Such a process would cause duplication of conditions and actions. This means that any source program obtained through a possible subsequent retranslation of decision tables to source code would have greater core storage requirements than the original. For this reason it was felt that such manipulations are better left under the direct control of the user through manipulation of the original source program or through the use of a proposed update facility discussed elsewhere in this paper.
2. If the table being merged is an action table, no other conditions need be satisfied to permit a merge. This is true as all other restrictions imposed (rules 3, 4, and 5) arise due to the presence of conditions in the tables under consideration.
3. The referring table must not be an action table. This restriction exists because, in order to merge two tables, there must be no intervening actions between the conditions of the referring table and those of the table referenced

in the rules in question. Since an action table contains no conditions but does contain actions, these are considered to be intervening and no merge can take place. Some representations of decision tables allow for an initialization section which would permit the above situation to be represented as a decision table with an initialization section consisting of the referring action table. The representation chosen in this system does not provide for an initialization section as it was felt that such a representation, although slightly more compact, leads to a loss of overall clarity.

4. There must be no actions intervening between the conditions of the referring table and those of the table referenced in the rules in question. As in rule 3 above, any such intervening actions may be considered as an initialization for the table referred to and this representation is not used in this system.
5. The resulting merged table must contain no more than forty rules due to the limitations imposed by the record formats. The reasons for this are discussed in chapter I of this paper.

c) Merge Algorithm

The algorithm used to determine whether a given table may be merged with the one referring to it is a straightforward application of the preceeding rules. It may be expressed as follows:

1. Read a table.
2. Read a table to which the previously read one refers.
3. Do any other tables refer to it? If so, we can do nothing but go to step 2 for another table.
4. Is the table an action table? If so, we can proceed to step 7 to merge.
5. Is the referring table an action table? If so, we can do nothing here. Go to step 2 for another table.
6. Does the rule which makes reference to the table under consideration contain any actions intervening between the conditions of the referring table and the conditions of the table referenced? If so, then we cannot merge and must return to step 2 for another table. If not, then we go to step 7 to merge the tables.
7. Will the resulting table have more than 40 rules? If not, we can merge, otherwise we cannot. We then return to step 2 for another table.

This algorithm can be expressed as the decision table given in fig. 10.

d) Operation

The procedure of combining tables once the above-mentioned criteria have been satisfied is a simple one. Action stubs of the receiving table are examined and the one containing the reference is isolated. The action entries are then examined to ascertain which rules contain the references. These rules are noted.

All condition records, if any, of the receiving

Read a table Read a table referenced	
Other tables refer to it?	YNNNNNN
Is it an action table?	YYNNNN
Is referring table action table?	YNNN
Intervening actions?	YNN
Result less than 40 rules?	YN YN
Merge	X X
Get another table	XXXXXXX
Repeat test	XXXXXXX

Fig. 10. Merge algorithm

table are now read into core. Condition entries for the rules previously noted are inserted as many times as there are rules in the table to be merged and condition entries of adjacent rules are shifted to the right by an equivalent number of positions.

Next, all condition records, if any, of the table being merged are read into core following those of the receiving table. Condition entries of these records are positioned in such a way as to form part of the rules originally noted. In order to chain the condition records together, the 'next record' pointer of the last condition record in the receiving table is altered to point to the first condition record of the other table. Similarly, the 'previous record' pointer of the first condition record in the table being merged is altered to point to the last condition record of the receiving table.

Once the condition records have been thus merged, all action records of the receiving table except that one containing the reference are read into core. The action entries are adjusted in the same manner as were the condition entries.

Finally, the action records of the table being merged are appended, their entry portions adjusted, and they are chained to those of the receiving table. The new condition and action records are then written out to the information file.

It has been stated that the action record containing

the reference to the merged table is not included in the resulting table. As this record is no longer required, it is chained to the end of the linked list of available records. Similarly, the 'name' and 'referenced by' records of the merged table are superfluous and so are also chained to the list of available records.

At this stage of the merge process the 'refers to' record of the resulting table must be created to reflect references of both of the original tables. Similarly, the 'referenced by' records of all tables previously referred to by the merged table must be updated. Once this has been done, the 'refers to' record of the merged table is released to the list of available records.

The last function which must be performed in order to complete the merge process is the update of the pointer record of the receiving table with the new counts of conditions, actions, and rules. Pointer records immediately preceding and following that of the absorbed table are chained together effectively deleting it.

This step completes the merge process and the next table is now tested for eligibility.

VI. THE PRINTER - DETBL30

a) Function

The function of this module is to produce, on request, a listing of the decision tables produced as well as a cross-reference list showing the inter-relations of these tables. Either one or both of these listings may be requested via the 'PRINT' control card.

b) Outputs

The decision table listing gives, for each table, the table name, all condition stubs and entries, and all action stubs and entries comprising the table. This information is printed in limited entry decision table format.

In addition to the system generated table name, the label of the last labelled statement encountered in the source program prior to the table name generation is shown. This permits the decision table to be more easily related back to the original source code.

The cross-reference list shows, for each table, names of all tables referring to it as well as all the tables to which it refers. This listing is useful for optimization and debugging purposes as it shows the inter-relations of the decision tables.

c) Operation

Listings are generated, one at a time, in two separate passes through the index file. The decision table list is generated during the first pass and the cross-reference during the second.

As the pointer record is read, the name, condition, and action records of the information file are accessed and the print lines of the table listing generated. On the second pass through the index file, the 'refers to' and 'referenced by' information records are read and the cross-reference listing is generated.

VII. THE PRINT UTILITY - DETBL90

a) Function

In order to facilitate control of listings produced as well as the maintenance of the system itself and the addition of possible future listings, it was decided to centralize all printing in one module.

A single file is generated containing print-image records for each of the requested listings. Each record is prefixed by a code which identifies it as forming part of a specific listing. It is then possible to extract the desired listings from this file and print them.

The function of this module is thus twofold: generation of the print file and the printing of all listings.

b) Operation

On initial entry to the system, this module is called from the monitor and the listing file is opened. Control is then returned to the monitor.

Generation of the file may now proceed. This is accomplished by means of calls to this module from the various other modules which generate listings. The print-image record with code attached is set up by the calling module and a call issued to entry point 'DETBL91'. The print utility then writes the record on the listing file and

returns control to the invoking module.

Once all control cards have been processed by the monitor it once again invokes the print utility, this time at entry point 'DETBL92'. All listings requested in the 'LIST' control card are now printed, one at a time. The module supplies page headings and controls page overflow based on the listing code.

VIII. IMPLEMENTATION AND RESULTS

a) Capabilities and Limitations

The DETABLS system described herein has been implemented on the CDC 6600 computer at Concordia University. In its present form it is capable of translating source programs written in the FORTRAN language into decision tables and combining these tables if certain rules are satisfied, as discussed in chapter V. Listings of the original source code, all non-executable statements except comments, decision tables, and decision table cross-references are produced on request. Appendix 'B' gives the outputs of six sample runs.

The system is at present somewhat inefficient. A FORTRAN program consisting of one hundred nineteen source cards was run through the system. The operations performed were the creation of decision tables, one merge pass, and printing of all listings. This was accomplished in eighteen seconds CPU time and required sixty thousand words of core memory. Although in this particular case forty one tables, which is a relatively large quantity, were merged, it is felt that system performance can be improved.

One method of improving performance is the use of the COMPASS language instead of COBOL. This would, however, make the system completely machine dependent which is contrary to the objective of producing a system which could be used

on any computer. It is felt that some improvements may be attained through other means.

The translator module makes extensive use of scanning input FORTRAN statements character by character and performs many character manipulations involving the use of subscripts. These operations are very costly in terms of CPU time. It is felt that performance would be improved through analysis and modification of the module to ensure that operations of this type are reduced to a minimum.

Further improvement of overall performance could be achieved by ensuring that use of subscripts for character by character scans as well as I/O of table records are reduced in the merge module. Both of these operations are used heavily and, it is felt, at times somewhat inefficiently in this module.

Two limitations are imposed by the system. The first is that the total number of rules in a decision table may not exceed forty. The second limitation sets a maximum of one hundred on the sum of conditions and actions in two tables to be merged. Neither of these limitations is considered serious as decision tables become too clumsy and confusing to work with long before they reach a size approaching these limits.

b) Control Cards

As implemented, the DETABLS system is driven by control cards of four distinct types. These are as follows:

1. *** CREATE = <language> - This control card invokes

the translator and formatter modules and causes the creation of the initial set of elementary decision tables. The only language supported at this time is FORTRAN.

2. *** LIST = <parameters> - This control card selects the listings to be produced in the run and must always appear first in the control card stream. Valid parameters are: SORC, NONX, TBLS, XREF or ALL which will yield the source, non-executable statements, decision table, cross-reference, or all listings respectively. If more than one parameter is coded, a non-blank character must separate it from its neighbor. If this card is incorrect, misplaced, or omitted, the option 'ALL' is assumed.
3. *** MERGE = <numeric integer> - This control card invokes the table merge module. It may be included at any time after the decision table files have actually been created. The card specifies the maximum number of passes to be made through the index file in attempting to merge tables. This figure must not exceed five.
4. *** PRINT = <parameters> - This control card invokes the printer module and selects the listings to be produced by this invocation. It may be specified at any time after the decision table files have been created and may be included any number of times. Valid parameters are: TBLS, XREF or ALL defined as on the 'LIST' control card described above. When more than one parameter is coded it must be separated from its neighbor by a non-blank character. In order for selected listings to be actually produced, they must also

have been specified on the 'LIST' control card.

The deck setup for a typical run is shown in fig. 11. This run will create decision tables from a FORTRAN source program, print them, pass once through the table merge module, and print the resulting merged tables as well as a cross-reference map.

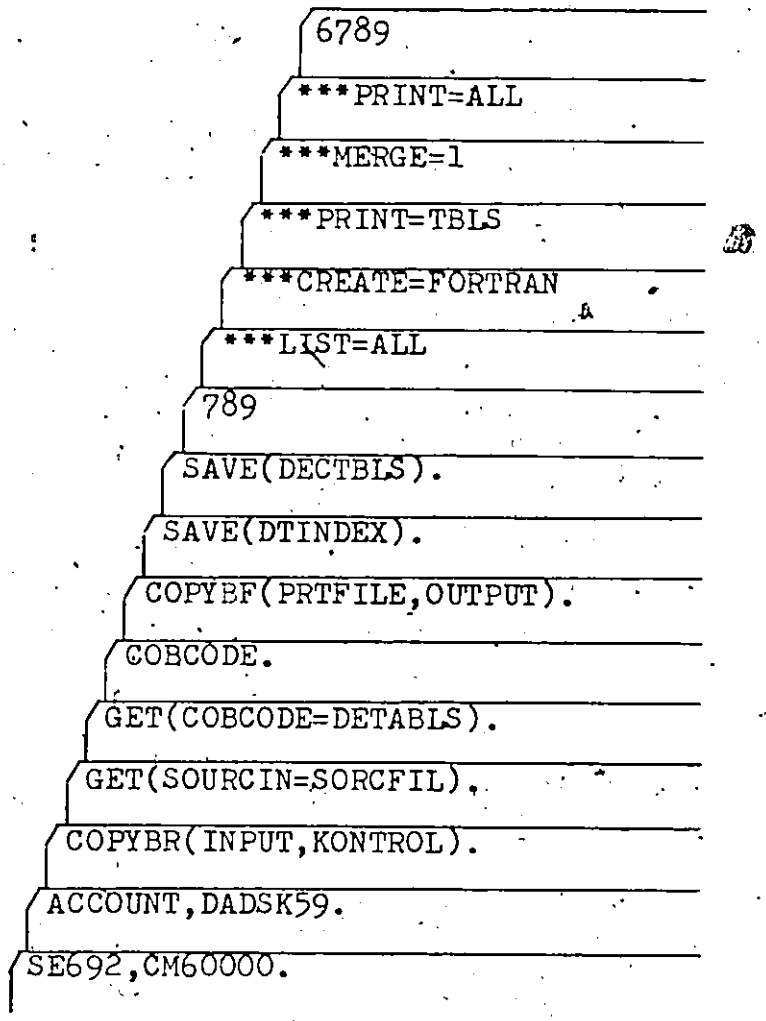


Fig. 11. Typical run deck set up

A source listing, decision table listing both before and after merge, and cross-reference listing are produced and the resulting table files saved for future use.

c) Analysis of Output

Appendix 'B' shows the DETABLS system output for six programs.

Program 1 contains three nested IF statements having the logical structure

IF --- THEN --- ELSE IF --- THEN ---

We see that such a logical structure produces one decision table and so the entire program translates into only two tables.

Examination of Table 1.00 reveals that some simplification could be achieved through introduction of the ELSE rule. In addition, if the system were to incorporate the initialization feature and the REPEAT TEST command the entire program could then be represented as one decision table. This table is shown in fig. 12.

The purpose of program 2 is to demonstrate the treatment of nested DO statements by the system. There are seven tables produced, six arising from the initialization of the DO loop variables and the assignment statement

TOTAL=K+IJ+L+M

and the seventh containing all the condition tests.

Examination of these tables reveals that Tables 0.00 through 0.05 could be combined into one if they were also replicated and merged with Table 1.01. This would yield a representation of the program using two tables only. Furthermore, if the representation of decision tables in the system allowed the use of an initialization section and the REPEAT TEST command were implemented, the entire

I=1	
J=2	
K=3	
I.GT.J	N
J.GT.K	N
K.GT.I	N
ELSE	Y
I=I+4	X
J=J+3	X
K=K+2	X
REPEAT TEST	X
PRINT 100,I,J,K	X
STOP	X
END	X

Fig. 12. Alternate table representation of program 1

program could then be represented by one decision table as shown in fig. 13.

Programs 3, 4, and 5 were originally intended to be logically identical with the physical differences that program 3 is written in a structured fashion, program 4 is not structured and program 5 uses the DO loop to replace some of the other FORTRAN statements. It was expected that logically identical decision tables would result from execution of the DETABLS system.

Examination of the results reveals that program 3 and 4 yield essentially identical decision tables. This result was expected as the programs are logically equivalent. An unexpected result however, was the difference in the tables produced from program 5. On investigation we see that the program is not actually equivalent to the other two and so it is reasonable that the decision tables produced would not be equivalent either.

This example points up the usefulness of this system for determining logic errors in programs. It confirms that differences in logic not immediately obvious from source code become much more striking when represented in decision table format.

The final example, program 6, consists of a small program forming part of an actual production system. Examination of the results shows that this program yields 28 decision tables. This seems to be a large number of tables to be derived from a program consisting of 119 source cards. The quantity of tables can be reduced manually by

I=1	
J=1	
IJ=5	
K=I+J	
L=5	
M=1	
TOTAL=K+IJ+L+M	
M=100	YYYYYN
L=10	YYYYN
IJ=100	YYYN
J=10	YYN
I=5	YN
I=I+1	X
J=1	X
J=J+2	X
IJ=5	XX
IJ=IJ+1	X
K=I+5	XXX
L=5	XXX
L=L+1	X
M=1	XXXX
M=M+1	X
TOTAL=K+IJ+L+M	XXXXX
REPEAT TEST	XXXXX
PRINT 100,TOTAL	X
STOP	X
END	X

Fig. 13. Alternate table representation of program 2

performing the following operations:

1. Tables can be replicated and merged with each table referring to them.
2. In cases where certain condition tests are independent of the actions immediately preceding them, the physical position of the actions may be changed so that they follow those conditions. This then permits further merging of tables.

The result of these operations yields 7 tables which completely describe program 6. These are given in appendix 'E'.

This substantial reduction in the number of tables required was possible under manual control due to the capability of determining the dependence or independence of certain actions and conditions and a subsequent modification of the program.

Theoretically, by means of further manipulations, it is possible ultimately to alter the entire program to produce a set of tables possessing a tree-like structure. This approach amounts to transformation of the original program into structured code.

The update facility discussed in chapter IX would permit the sort of manipulation performed here. Also discussed in chapter IX is the automation of some of these operations.

Further examination of the output for program 6 reveals some weaknesses in the DETABLS system. One such is encountered in Tables 0.01 and 5.01 as well as others. The action stub 'GO TO ---' appears more than once in the

same table. Although this is not technically incorrect, it is confusing to someone examining the table output. Such an action stub should only appear once in any one table.

Another weakness of the system is the presence of action tables containing only one action, a 'GO TO ---' stub. Table 9.00 is an example of such a table. These tables are entirely unnecessary and the system should be modified to alter the original reference and eliminate these tables whenever they are encountered.

The handling of comment statements is a problem. In the present version of the system, comments remain in the generated decision tables and appear on the decision table listing as if they were executable statements or a part of the table name. This treatment can, however, lead to confusion as comments included in a particular source program are not necessarily logically connected with the immediately preceding or following statements. Thus their inclusion in a decision table could, under certain circumstances, lead to their being entirely removed from the items which they should clarify by virtue of their being in a physically separate table. The problem is further complicated by the table manipulations and transformations which take place.

The decision table listing of program 6 points out these problems. Table 6.00 contains a comment referring to 'THE ABOVE' which has now become meaningless. Similarly Table 39.00 of the same listing contains a comment which implies that the table describes a routine for printing

which it does not. It contains merely a reference to a table which then describes the print routine.

It is felt that the solution to the problem is to treat comment statements in the same way as all other non-executable statements and include them in the list of non-execuatble statements or perhaps a separate listing altogether. They should not appear as an integral part of any decision table.

d) Listings

The listings produced and shown in appendix 'B' are quite clear, however, the true value of the list of non-executable statements is questionable. Its only function is to show which source statements have not been included in the generated decision tables. Thus, together with the decision table listing, it makes available all original source statements.

The usefulness of the system could be enhanced through the addition of three other listings. Sorted listings of all different condition and action stubs together with the names of all tables in which they appear would assist the user in analyzing his logic.

He could quickly determine if any condition tests were missing or unnecessary. Any unnecessary duplication of condition tests and actions would also be quickly noticed. It would then be possible for the user to alter the source code depending on his constraints of core and time available to yield a more satisfactory program.

In addition to the above two listings, a summary statistics report giving such information as the number of cards read, source statements processed, total number of conditions and actions, and other statistics would be desirable. Addition of such a report would involve a minimal effort.

IX. SOME EXTENSIONS

As has been mentioned earlier in this paper, this system was specifically designed in a modular fashion to permit the easy addition of new features and enhancements. This section deals briefly with some of the potential additions to the system.

a) Other Languages

It is possible to implement the processing of programs witten in other languages simply by writing a separate translator for each language desired. This translator would then be invoked through the appropriate 'CREATE' control card.

An alternative to this approach is to develop a generalized translator able to handle programs written in any language. Such a translator would be driven by a language description written in a Metalanguage such as Sherman's [11] STRIP.

A similar technique is used in the generation of compiler compilers such as the 'NEST' (Northern Electric Syntax-directed Translator) system developed by Northern Electric Co. Ltd.

b) Structured Programming

The control logic of any program can be represented by the control structures

IF THEN - ELSE

DO WHILE

simple sequencing

This theorem is due to C. Bohm and G. Jacopini [1]. Thus it is possible to transform any program into a 'GO TO'-less program using the above control structures. Given the additional facility of a macro call via the 'PERFORM' construct, it is possible to gather together the most important statements in the program and to 'PERFORM' the missing code. This process will yield a program skeleton and may be repeated at various levels to give as broad or as detailed a description of the program as desired.

A potential enhancement to the DETABLS system would then be the implementation of a module which could operate on the elementary decision tables produced by the formatter module and produce structured code.

c) Update Facility

One of the primary functions fulfilled by this system is to provide an aid to program analysis and optimization. This implies that the source program under study would probably be under development and therefore subject to change. For this reason, another possible enhancement which could prove useful would be one which would permit the direct update of decision tables.

The facility of manipulating the entire decision table and parts thereof directly rather than modifying the source code could greatly speed up the debugging process as it is simpler to work at the decision table level rather than with source code. Such a feature could be incorporated into the system by the addition of an update module. This would provide the user with direct control of the following functions:

1. Deletion of specified tables.
2. Addition of tables.
3. Alteration, addition, and deletion of individual condition and action stubs and entries.
4. Alteration of the name record associated with each table.
5. Merge of two tables even though rules 1 and 4 for merge, described in chapter V are not satisfied. This merge would not cause deletion of any tables but rather would replicate the table being merged.

Maintenance of the chain link pointers connecting the individual records of the index and information files would be performed automatically by the update module as would the maintenance of the REFERS TO and REFERENCED BY lists associated with the affected tables.

Implementation of an update module capable of performing the above functions would facilitate a further enhancement of the system -- interactive operation. This could be accomplished very simply by modifying the monitor module,

DETBLOO, to accept all control commands from an interactive terminal rather than a card file. All or selected output could also be performed on this terminal.

The interactive mode would allow the user more immediate access to the results of his manipulations thereby speeding up the process, however, at the expense of increased overhead with respect to the batch mode.

Operation in a batch mode mode would be simpler and less expensive to use but the time taken to complete the debugging process would be considerably longer in this case due to a much longer turnaround time.

Thus the choice between batch or interactive operation must be made on an individual basis by each user based on his requirements and priorities.

d) Retranslation

Systems are currently available for the purpose of translating decision tables to nearly optimum programs. One such system is the IBM 360 Decision Logic Translator which translates decision tables into a FORTRAN source program.

The addition of such a feature to the DETABLS system would permit the user to break down his original source code to decision tables, operate on them to optimize the logic according to his constraints, and then reconstruct his program in the original source language. Used in this manner, the DETABLS system would become a very valuable program optimization tool indeed.

As with the implementation of forward translators

for other languages, the system could remain language-independent by providing separate modules for each language to be supported or a generalized translator driven by a Metalanguage.

e) Variable Analysis

It has previously been mentioned that two tables will not be combined if the rule containing the reference to the table to be merged contains other action entries. This restriction was imposed to ensure that tables will not be merged if the action stubs of the receiving table and the condition stubs of the table to be merged are not independent.

A solution to this problem could be attempted by performing a 'variable analysis' to determine if the variables tested in the condition stubs of the table to be merged were affected by the variables altered in the action stubs of the receiving table.

This solution, however, is not entirely satisfactory as it is not always possible to determine whether two variables are truly independent or not. It is entirely possible that the relation of two variables may depend on the values of certain data items input to the program. Thus their dependence or independence would be determined at run time. It is also possible to write code in which variables which logically appear independent are really not so due to the hardware configuration of the computer in question. In these and many other cases, the dependence or independence of two variables cannot be ascertained

from the source code.

As a result of this uncertainty, a merging of tables based on a variable analysis could yield incorrect results. This approach would be useful, however, to produce a listing of variables and presumed table interactions. Based on this output, the user could then modify his code as required to obtain a better program.

X. CONCLUSIONS

The original objective was to produce a system capable of the translation of source code to decision tables and which would generate output suitable for documentation and program analysis. It is felt that these objectives have been met.

The source listing and associated decision table and cross-reference listings provide excellent program documentation. As expected, program logic is more readily understood from the decision table listing than from the source listing. Flaws in logic are much more readily observed on examination of the decision table and cross-reference listings.

Modularity of the system and the basic language independence inherent in the design make it suitable for the implementation of additional features as previously discussed. Thus the system in its present state could serve as the basis for a more comprehensive generalized translator and decision table manipulator.

APPENDIX A

LIST OF FORTRAN STATEMENTS BY CATEGORY

1. Comments - Any statement containing the characters 'C', '$', '*', or'.' in column 1.

2. Non-executable -
   - ENTRY
   - PROGRAM
   - SUBROUTINE
   - FUNCTION
   - EXTERNAL
   - COMPLEX
   - DOUBLE PRECISION
   - DOUBLE
   - REAL
   - INTEGER
   - LOGICAL
   - CHARACTER
   - TYPE
   - DIMENSION
   - COMMON
   - EQUIVALENCE
   - DATA
   - BLOCK DATA
   - FORMAT

NAMELIST

Any statement containing the character 'F' in column 1.

3. Conditional -

GO TO m,(n_1,....,n_m)

GO TO (n_1,....,n_m),i

IF

DO

4. Unconditional -

CALL

RETURN

Assignment statements

Arithmetic statement function

GO TO

ASSIGN

CONTINUE

PAUSE

SENSE LIGHT

STOP

BUFFER IN

BUFFER OUT

DECODE

ENCODE

PRINT

PUNCH

READ

WRITE

REWIND

```
BACKSPACE
END
```

APPENDIX B

SAMPLE RUNS OF DETABLS SYSTEM

PROGRAM 1

```
      FL REQUIRED TO LOAD    45500
      FL REQUIRED TO RUN     44600
* DET4005 --- FORTRAN TRANSLATION REQUESTED
* DET1010 --- TABLE GENERATION BEGINS
* DET2006 --- TABLE MERGE REQUESTED
* DET2009 --- MERGE PHASE PASS NUMBER ---    0001
* DET2008 --- COMBINING TABLES ---            1.01 AND   1.00
* DET2008 --- COMBINING TABLES ---            1.02 AND   1.00
* DET2008 --- COMBINING TABLES ---            1.03 AND   1.00
* DET2008 --- COMBINING TABLES ---            2.00 AND   1.00
* DET2008 --- COMBINING TABLES ---            3.00 AND   1.00
* DET3003 --- DECISION TABLE PRINT REQUESTED
* DET0002 --- JOB TERMINATED SUCCESSFULLY
```

```
      PROGRAM SHOWIF
      I=1
      J=2
      K=3
    5 IF(I.GT.J) GO TO 10
      IF(J.GT.K) GO TO 10
      IF(K.GT.I) GO TO 10
      GO TO 15
   10 I=I+4
      J=J+3
      K=K+2
      GO TO 5
   15 PRINT 100,I,J,K
  100 FORMAT(3I5)
      STOP
      END
```

```
   PROGRAM SHOWIF
100FORMAT(3I5)
```

```
*** TABLE 000.00 --- RELATED TO ORIGINAL LABEL 00000
CONDITIONS =  0   ACTIONS =  4   RULES =  1

I=1                                                  I X
J=2                                                  I X
K=3                                                  I X
GO TO 001.00                                         I X
```

```
*** TABLE 001.00 --- RELATED TO ORIGINAL LABEL 00005
CONDITIONS =  3   ACTIONS =  7   RULES =  4

I.GT.J                                               I YNNN
J.GT.K                                               I  YNN
K.GT.I                                               I   YN.
-------------------------------------------------------- I ----------------------------------------
I=I+4                                                I XXX
J=J+3                                                I XXX
K=K+2                                                I XXX
GO TO 001.00                                         I XXX
PRINT 100,I,J,K                                      I    X
STOP                                                 I    X
END                                                  I    X
```

```
000.00  REFERS TO      001.00
        REFERENCED BY FIRST

001.00  REFERS TO      001.00  000.00
        REFERENCED BY 000.00  001.00
```

PROGRAM 2

```
        FL REQUIRED TO LOAD   45500
        FL REQUIRED TO RUN    44600
*** DET4005 --- FORTRAN TRANSLATION REQUESTED
*** DET1010 --- TABLE GENERATION BEGINS
*** DET2006 --- TABLE MERGE REQUESTED
*** DET2009 --- MERGE PHASE PASS NUMBER ---    0001
*** DET2008 --- COMBINING TABLES ---            1.00 AND   0.05
*** DET2008 --- COMBINING TABLES ---            2.00 AND   1.01
*** DET2008 --- COMBINING TABLES ---            2.01 AND   1.01
*** DET2008 --- COMBINING TABLES ---            3.00 AND   1.01
*** DET2008 --- COMBINING TABLES ---            3.01 AND   1.01
*** DET2008 --- COMBINING TABLES ---            4.00 AND   1.01
*** DET2008 --- COMBINING TABLES ---            4.01 AND   1.01
*** DET2008 --- COMBINING TABLES ---            5.00 AND   1.01
*** DET2008 --- COMBINING TABLES ---            5.01 AND   1.01
*** DET2008 --- COMBINING TABLES ---            5.02 AND   1.01
*** DET3003 --- DECISION TABLE PRINT REQUESTED
*** DET0002 --- JOB TERMINATED SUCCESSFULLY
```

```
      PROGRAM NESTDO
      DO 50 I=1,5
      DO 40 J=1,10,2
      DO 30 IJ=5,100
      K=I+J
      DO 20 L=5,10
      DO 10 M=1,100
      TOTAL=K+IJ+L+M
   10 CONTINUE
   20 CONTINUE
   30 CONTINUE
   40 CONTINUE
   50 CONTINUE
      PRINT 100,TOTAL
  100 FORMAT(F5.1)
      STOP
      END
```

```
  PROGRAM NESTDO
100FORMAT(F5.1)
```

```
*** TABLE 000,00 *** RELATED TO ORIGINAL LABEL 00000
CONDITIONS =  0   ACTIONS =  2   RULES =  1

I=1                                                    I X
GO TO 000,01                                           I X

*** TABLE 000,01 *** RELATED TO ORIGINAL LABEL 00000
CONDITIONS =  0   ACTIONS =  2   RULES =  1

J=1                                                    I X
GO TO 000,02                                           I X

*** TABLE 000,02 *** RELATED TO ORIGINAL LABEL 00000
CONDITIONS =  0   ACTIONS =  2   RULES =  1

IJ=5                                                   I X
GO TO 000,03                                           I X

*** TABLE 000,03 *** RELATED TO ORIGINAL LABEL 00000
CONDITIONS =  0   ACTIONS =  3   RULES =  1

K=I+J                                                  I X
L=5                                                    I X
GO TO 000,04                                           I X

*** TABLE 000,04 *** RELATED TO ORIGINAL LABEL 00000
CONDITIONS =  0   ACTIONS =  2   RULES =  1

M=I                                                    I X
GO TO 000,05                                           I X
```

```
*** TABLE 000,05 --- RELATED TO ORIGINAL LABEL 00000
CONDITIONS =  0   ACTIONS =  2   RULES =  1

TOTAL=K+IJ+L+M                                  I X
GO TO 001,01                                    I X
```

```
*** TABLE 001,01 --- RELATED TO ORIGINAL LABEL 00010
CONDITIONS =  5   ACTIONS = 13   RULES =  6

M=100                                           I YYYYYN
L=10                                            I YYYYN
IJ=100                                          I YYYN
J=10                                            I YYN
I=5                                             I YN
                                                I ------
M=M+1                                           I     X
GO TO 000,05                                    I     X
L=L+1                                           I    X
GO TO 000,04                                    I    X
IJ=IJ+1                                         I   X
GO TO 000,03                                    I   X
J=J+2                                           I  X
GO TO 000,02                                    I  X
I=I+1                                           I X
GO TO 000,01                                    I X
PRINT 100,TOTAL                                 I X
STOP                                            I X
END                                             I X
```

```
000.00  REFERS TO      000.01
        REFERENCED BY FIRST

000.01  REFERS TO      000.02
        REFERENCED BY 000.00  001.01

000.02  REFERS TO      000.03
        REFERENCED BY 000.01  001.01

000.03  REFERS TO      000.04
        REFERENCED BY 000.02  001.01

000.04  REFERS TO      000.05
        REFERENCED BY 000.03  001.01

000.05  REFERS TO      001.01
        REFERENCED BY 000.04  001.01

001.01  REFERS TO      000.05  000.04  000.03  000.02  000.01  000.00
        REFERENCED BY 000.05
```

PROGRAM 3

```
        FL REQUIRED TO LOAD   45500
        FL REQUIRED TO RUN    44600
*** DET4005 --- FORTRAN TRANSLATION REQUESTED
*** DET1010 --- TABLE GENERATION BEGINS
*** DET2006 --- TABLE MERGE REQUESTED
*** DET2009 --- MERGE PHASE PASS NUMBER ---    0001
*** DET2008 --- COMBINING TABLES ---            1.00 AND   2.00
*** DET2008 --- COMBINING TABLES ---            2.01 AND   2.00
*** DET2008 --- COMBINING TABLES ---            3.00 AND   4.00
*** DET2008 --- COMBINING TABLES ---            3.01 AND   4.00
*** DET2008 --- COMBINING TABLES ---            3.02 AND   4.00
*** DET2008 --- COMBINING TABLES ---            4.01 AND   4.00
*** DET2008 --- COMBINING TABLES ---            4.03 AND   4.02
*** DET3003 --- DECISION TABLE PRINT REQUESTED
*** DET0002 --- JOB TERMINATED SUCCESSFULLY
```

```
      PROGRAM STRCTRD
      DIMENSION NEV(4),POS(4)
      IJ=1
      K=5
      GO TO 10
    5 CONTINUE
      POS(IJ)=0
      IJ=IJ+1
   10 IF(IJ.LE.K) GO TO 5
      IZ=2
      MIN=NEV(1)
      GO TO 20
   15 CONTINUE
      IF(NEV(IZ).LE.MIN) MIN=NEV(IZ)
      IZ=IZ+1
   20 IF(IZ.LE.5) GO TO 15
      IF(MIN.EQ.NEV(1)) POS(1)=1
      IF(MIN.EQ.NEV(2)) POS(2)=1
      STOP
      END
```

```
PROGRAM STRCTRD
DIMENSION NEV(4),POS(4)
```

```
*** TABLE 000.00 --- RELATED TO ORIGINAL LABEL 00000
CONDITIONS = 0   ACTIONS = 3   RULES = 1

IJ=1                                                    I X
K=5                                                     I X
GO TO 002.00                                            I X

*** TABLE 002.00 --- RELATED TO ORIGINAL LABEL 00010
CONDITIONS = 1   ACTIONS = 6   RULES = 2

IJ.LE.K                                                 I YN
                                                        I --
POS(IJ)=0                                               I X
IJ=IJ+1                                                 I X
GO TO 002.00                                            I X
IZ=2                                                    I  X
MIN=NEV(1)                                              I  X
GO TO 004.00                                            I  X

*** TABLE 004.00 --- RELATED TO ORIGINAL LABEL 00020
CONDITIONS = 3   ACTIONS = 6   RULES = 4

IZ.LE.5                                                 I YYNN
NEV(IZ).LE.MIN                                          I YN
MIN.EQ.NEV(1)                                           I   YN
                                                        I --
MIN=NEV(IZ)                                             I X
IZ=IZ+1                                                 I XX
GO TO 004.00                                            I XX
POS(1)=1                                                I   X
GO TO 004.02                                            I   X
GO TO 004.02                                            I    X

*** TABLE 004.02 --- RELATED TO ORIGINAL LABEL 00020
CONDITIONS = 1   ACTIONS = 3   RULES = 2

MIN.EQ.NEV(2)                                           I YN
                                                        I --
POS(2)=1                                                I X
STOP                                                    I XX
END                                                     I XX
```

```
** DECISION TABLE GENERATOR RUN **   ** CROSS REFERENCE **   PAGE    1

000.00  REFERS TO       002.00
        REFERENCED BY FIRST

002.00  REFERS TO       002.00  004.00
        REFERENCED BY 000.00  002.00

004.00  REFERS TO       004.00  004.02
        REFERENCED BY 002.00  004.00

004.02  REFERS TO       NO TABLES *****
        REFERENCED BY 004.00
```

PROGRAM 4

```
      FL REQUIRED TO LOAD   45500
      FL REQUIRED TO RUN    44600
** DET4005 --- FORTRAN TRANSLATION REQUESTED
** DET1010 --- TABLE GENERATION BEGINS
** DET2006 --- TABLE MERGE REQUESTED
** DET2009 --- MERGE PHASE PASS NUMBER ---    0001
** DET2008 --- COMBINING TABLES ---            1.01 AND   1.00
** DET2008 --- COMBINING TABLES ---            2.00 AND   1.00
** DET2008 --- COMBINING TABLES ---            3.01 AND   3.00
** DET2008 --- COMBINING TABLES ---            4.01 AND   4.00
** DET2008 --- COMBINING TABLES ---            5.00 AND   3.00
** DET2008 --- COMBINING TABLES ---            6.00 AND   4.00
** DET2008 --- COMBINING TABLES ---            7.00 AND   3.00
** DET2008 --- COMBINING TABLES ---            7.01 AND   3.00
** DET2008 --- COMBINING TABLES ---            8.00 AND   3.00
** DET2008 --- COMBINING TABLES ---            9.00 AND   3.00
** DET2008 --- COMBINING TABLES ---           10.00 AND   4.00
** DET3003 --- DECISION TABLE PRINT REQUESTED
** DET0002 --- JOB TERMINATED SUCCESSFULLY
```

```
      PROGRAM UNSTRCD
      DIMENSION NEV(4),POS(4)
      IJ=1
      K=5
    5 IF(IJ.LE.K) GO TO 10
      IZ=2
      MIN=NEV(1)
      GO TO 15
   10 POS(IJ)=0
      IJ=IJ+1
      GO TO 5
   15 IF(IZ.LE.5) GO TO 35
      IF(MIN.EQ.NEV(1)) GO TO 25
   20 IF(MIN.EQ.NEV(2)) GO TO 30
      GO TO 99
   25 POS(1)=1
      GO TO 20
   30 POS(2)=1
      GO TO 99
   35 IF(NEV(IZ).LE.MIN) GO TO 40
      GO TO 45
   40 MIN=NEV(IZ)
   45 IZ=IZ+1
      GO TO 15
   99 STOP
      END
```

```
PROGRAM UNSTRCD
DIMENSION NEV(4),POS(4)
```

```
*** TABLE 000,00 --- RELATED TO ORIGINAL LABEL 00000
CONDITIONS =  0   ACTIONS =  3   RULES =  1
```

IJ=1	X
K=5	X
GO TO 001,00	X

```
*** TABLE 001,00 --- RELATED TO ORIGINAL LABEL 00005
CONDITIONS =  1   ACTIONS =  6   RULES =  2
```

IJ.LE.K	YN
IZ=2	X
MIN=NEV(1)	X
GO TO 003,00	X
POS(IJ)=0	X
IJ=IJ+1	X
GO TO 001,00	X

```
*** TABLE 003,00 --- RELATED TO ORIGINAL LABEL 00015
CONDITIONS =  3   ACTIONS =  6   RULES =  4
```

IZ.LE.5	YYNN
MIN.EQ.NEV(1)	YN
NEV(IZ).LE.MIN	YN
GO TO 004,00	X
POS(1)=1	X
GO TO 004,00	X
MIN=NEV(IZ)	X
IZ=IZ+1	XX
GO TO 003,00	XX

```
*** TABLE 004,00 --- RELATED TO ORIGINAL LABEL 00020
CONDITIONS =  1   ACTIONS =  3   RULES =  2
```

MIN.EQ.NEV(2)	YN
POS(2)=1	X
STOP	XX
END	XX

```
000.00  REFERS TO      001.00
        REFERENCED BY FIRST

001.00  REFERS TO      003.00  001.00
        REFERENCED BY 000.00  001.00

003.00  REFERS TO      004.00  003.00
        REFERENCED BY 001.00  003.00

004.00  REFERS TO       NO TABLES *****
        REFERENCED BY 003.00
```

PROGRAM 5

```
      FL REQUIRED TO LOAD   45500
      FL REQUIRED TO RUN    44600
** DET4005 --- FORTRAN TRANSLATION REQUESTED
** DET1010 --- TABLE GENERATION BEGINS
** DET2006 --- TABLE MERGE REQUESTED
** DET2009 --- MERGE PHASE PASS NUMBER ---   0001
** DET2008 --- COMBINING TABLES ---          1.00 AND   0.01
** DET2008 --- COMBINING TABLES ---          1.02 AND   1.01
** DET2008 --- COMBINING TABLES ---          1.04 AND   1.03
** DET2008 --- COMBINING TABLES ---          2.00 AND   1.03
** DET2008 --- COMBINING TABLES ---          2.02 AND   2.01
** DET2008 --- COMBINING TABLES ---          2.04 AND   2.03
** DET3003 --- DECISION TABLE PRINT REQUESTED
** DET0002 --- JOB TERMINATED SUCCESSFULLY
```

```
      PROGRAM USEDO
      DIMENSION NEV(4),POS(4)
      K=5
      DO 5 IJ=1,K
      POS(IJ)=0
    5 CONTINUE
      MIN=NEV(1)
      DO 20 IZ=2,5
      IF(NEV(IZ).GT.MIN) GO TO 20
      MIN=NEV(IZ)
   20 CONTINUE
      IF(MIN.EQ.NEV(1)) POS(1)=1
      IF(MIN.EQ.NEV(2)) POS(2)=1
      STOP
      END
```

```
PROGRAM USEDO
DIMENSION NEV(4),POS(4)
```

```
I *** TABLE 000,00 --- RELATED TO ORIGINAL LABEL 00000                    I
I CONDITIONS =  0   ACTIONS =  3   RULES =  1                             I
I                                                                         I
I-------------------------------------------------------------------------I
I K=5                                                        I X          I
I IJ=1                                                       I X          I
I GO TO 000,01                                               I X          I
```

```
I *** TABLE 000,01 --- RELATED TO ORIGINAL LABEL 00000                    I
I CONDITIONS =  0   ACTIONS =  2   RULES =  1                             I
I                                                                         I
I-------------------------------------------------------------------------I
I POS(IJ)=0                                                  I X          I
I GO TO 001,01                                               I X          I
```

```
I *** TABLE 001,01 --- RELATED TO ORIGINAL LABEL 00005                    I
I CONDITIONS =  1   ACTIONS =  5   RULES =  2                             I
I                                                                         I
I-------------------------------------------------------------------------I
I IJ=K                                                       I YN         I
I ---------------------------------------------------------- I ---------- I
I IJ=IJ+1                                                    I  X         I
I GO TO 000,01                                               I  X         I
I MIN=NEV(1)                                                 I X          I
I IZ=2                                                       I X          I
I GO TO 001,03                                               I X          I
```

```
I *** TABLE 001,03 --- RELATED TO ORIGINAL LABEL 00005                    I
I CONDITIONS =  1   ACTIONS =  2   RULES =  2                             I
I                                                                         I
I-------------------------------------------------------------------------I
I NEV(IZ).GT.MIN                                             I YN         I
I ---------------------------------------------------------- I ---------- I
I MIN=NEV(IZ)                                                I X          I
I GO TO 002,01                                               I XX         I
```

*** TABLE 002,01 --- RELATED TO ORIGINAL LABEL 00020
CONDITIONS = 2 ACTIONS = 5 RULES = 3

IZ=5	YYN
MIN,EQ,NEV(1)	YN
IZ=IZ+1	
GO TO 001,03	X
POS(1)=1	X
GO TO 002,03	X
GO TO 002,03	X

*** TABLE 002,03 --- RELATED TO ORIGINAL LABEL 00020
CONDITIONS = 1 ACTIONS = 3 RULES = 2

MIN,EQ,NEV(2)	YN
POS(2)=1	X
STOP	XX
END	XX

```
000.00  REFERS TO      000.01
        REFERENCED BY FIRST

000.01  REFERS TO      001.01
        REFERENCED BY 000.00  001.01

001.01  REFERS TO      000.01  001.03
        REFERENCED BY 000.01

001.03  REFERS TO      002.01
        REFERENCED BY 001.01  002.01

002.01  REFERS TO      001.03  002.03
        REFERENCED BY 001.03

002.03  REFERS TO      NO TABLES *****
        REFERENCED BY 002.01
```

PROGRAM 6

```
DET4005 --- FORTRAN TRANSLATION REQUESTED
DET1010 --- TABLE GENERATION BEGINS
DET2006 --- TABLE MERGE REQUESTED
DET2009 --- MERGE PHASE PASS NUMBER ---    0001
DET2008 --- COMBINING TABLES ---            0.02 AND    0.01
DET2008 --- COMBINING TABLES ---            0.03 AND    0.01
DET2008 --- COMBINING TABLES ---            1.00 AND    0.01
DET2008 --- COMBINING TABLES ---            1.01 AND    0.01
DET2008 --- COMBINING TABLES ---            2.00 AND    0.01
DET2008 --- COMBINING TABLES ---            3.00 AND    0.01
DET2008 --- COMBINING TABLES ---            4.00 AND    0.01
DET2008 --- COMBINING TABLES ---            5.02 AND    5.01
DET2008 --- COMBINING TABLES ---            8.00 AND    7.00
DET2008 --- COMBINING TABLES ---            9.02 AND    9.01
DET2008 --- COMBINING TABLES ---           10.00 AND    9.01
DET2008 --- COMBINING TABLES ---           11.00 AND    9.01
DET2008 --- COMBINING TABLES ---           12.00 AND    9.01
DET2008 --- COMBINING TABLES ---           14.00 AND   13.00
DET2008 --- COMBINING TABLES ---           15.00 AND    9.01
DET2008 --- COMBINING TABLES ---           17.00 AND   16.00
DET2008 --- COMBINING TABLES ---           18.00 AND   16.00
DET2008 --- COMBINING TABLES ---           19.00 AND   16.00
DET2008 --- COMBINING TABLES ---           20.00 AND   13.00
DET2008 --- COMBINING TABLES ---           21.00 AND    5.01
DET2008 --- COMBINING TABLES ---           21.01 AND    5.01
DET2008 --- COMBINING TABLES ---           22.00 AND    5.01
DET2008 --- COMBINING TABLES ---           23.00 AND   22.01
DET2008 --- COMBINING TABLES ---           24.00 AND   23.02
DET2008 --- COMBINING TABLES ---           24.01 AND   23.02
DET2008 --- COMBINING TABLES ---           24.02 AND   23.02
DET2008 --- COMBINING TABLES ---           25.00 AND   23.02
DET2008 --- COMBINING TABLES ---           26.00 AND   23.02
DET2008 --- COMBINING TABLES ---           27.00 AND   23.02
DET2008 --- COMBINING TABLES ---           28.00 AND   23.02
DET2008 --- COMBINING TABLES ---           29.00 AND   28.01
DET2008 --- COMBINING TABLES ---           30.00 AND   23.02
DET2008 --- COMBINING TABLES ---           31.00 AND   23.02
DET2008 --- COMBINING TABLES ---           33.00 AND   32.00
DET2008 --- COMBINING TABLES ---           34.00 AND   23.02
DET2008 --- COMBINING TABLES ---           35.00 AND   34.01
DET2008 --- COMBINING TABLES ---           36.02 AND   36.01
DET2008 --- COMBINING TABLES ---           37.00 AND   23.02
DET2008 --- COMBINING TABLES ---           40.00 AND   39.01
DET2008 --- COMBINING TABLES ---           40.02 AND   40.01
DET2008 --- COMBINING TABLES ---           41.00 AND    5.01
DET3003 --- DECISION TABLE PRINT REQUESTED
DET0002 --- JOB TERMINATED SUCCESSFULLY
```

```
      PROGRAM TESTER2
      CHARACTER TEMP,NCBLANK,J1,NTHD,NU2H
      DIMENSION STO(8)
      COMMON IOFCO(3),NUMF(3,7),NUMT(3,7),NTCOF(3),NTCOT(3),
     1NRCO(3),IVAR(50),TEMP(58),I4,JVAR(50),NTHD(5,5),NCBLANK,
     2NU2H(5,5),NBLANK,NTR,ME3,ME4,ME5
      NBLANK=(4H    )
      NCBLANK=(1H )
      KNU=ME1=ME2=NT1=NT2=0
      ASSIGN 10 TO LABEL
      DUMVAR=1
      GO TO LABEL,(17,10,4)
   17 IF(DUMVAR)2,3,4
    2 DUMVAR=2
      GO TO 30
    3 DUMVAR=3
      GO TO 20
    4 DUMVAR=4
      I2345=1$J2345= 2 $ K2345 = 6
   10 READ(60,201)XJOB,ICAR,STO
  201 FORMAT(A6,A3,7A8,A7)
      GO TO (160,20)EOFCKF(60)
   20 KNU=KNU+1
**********THE ABOVE ALREADY EXISTS**********

         READING OF #NUM# CARDS AND INFORMATION STORAGE

      J1=12B
      DECODE(63,202,STO)IOFCO(KNU),(TEMP(I),I=1,58)
  202 FORMAT(1X,A4,58A1)
      I1=0
      DO 80 I=1,7
      I3=0
      DO 50 J=1,8
      I1=I1+1
      IF(TEMP(I1).GE.J1)30,50
   30 IF(TEMP(I1).EQ.NCBLANK)40,90
   40 I3=I3+1
   50 CONTINUE
      IF(I3.GT.0.AND.I3.LT.8)90,60
   60 IF(I.EQ.7)110,70
   70 DECODE(13,203,STO(I))NUMF(KNU,I),NUMT(KNU,I)
  203 FORMAT(5X,2A4)
   80 CONTINUE
   90 IF(I.EQ.7)100,150
  100 NT1=KNU
      NTCOF(KNU)=NTCOT(KNU)=NBLANK
      GO TO 120
  110 DECODE(13,203,STO(I))NTCOF(KNU),NTCOT(KNU)
  120 IF(TEMP(57).EQ.NCBLANK.AND.TEMP(58).EQ.NCBLANK)130,125
  125 IF(TEMP(57).LT.J1.AND.TEMP(58).LT.J1)130,140
  130 DECODE(7,204,STO(8))NRCO(KNU)
  204 FORMAT(5X,A2)
      GO TO 10
  140 ME2=KNU
      GO TO 10
```

```
  150 ME1=KNU
      GO TO 10
  160 CONTINUE
C
C
C           TRANSLATOR CALCULATION AND OFFICE CODE DETERMINATION
C
      IF(ME1.EQ.0.AND.ME2.EQ.0)300,310
  300 I1=I3=1
      I2=NTR=0
      IF(KNU.EQ.1)290,170
  170 DO 280 J=1,KNU
      DECODE(3,205,IOFCO(J))(TEMP(I),I=1,3)
  205 FORMAT(3A1)
      DO 180 I=1,3
      IF(TEMP(I).EQ.NCBLANK)260,180
  180 CONTINUE
      IF(I2.EQ.1)190,230
  190 IF(IOFCO(J).EQ.IOFCO(J-1))280,191
  191 IF((J+1).GT.KNU)200,192
  192 IF(IOFCO(J+1).EQ.IOFCO(1))220,200
  200 CALL TRASU(I1,J-I,I3)
      IF(ME3.NE.0.OR.ME4.NE.0.OR.ME5.NE.0)860,870
  870 I1=J
      GO TO 280
  220 I1=I3=2
      GO TO 320
  230 I2=J
  240 IF(I2.EQ.1)280,250
  250 I2=I2-1
      IOFCO(I2)=IOFCO(J)
      GO TO 240
  260 NT2=1
      IF(I2.EQ.1)270,280
  270 IOFCO(J)=IOFCO(J-1)
  280 CONTINUE
      GO TO 290
  320 CALL TRASU(1,3,2)
      IF(ME3.NE.0.OR.ME4.NE.0.OR.ME5.NE.0)860,290
  290 CALL TRASU(I1,KNU,I3)
  860 CONTINUE
C
C
C                   PRINTING
C
      DO 330 L=1,NTR
      J2=L-1
      PRINT 111,J2
  111 FORMAT(12H TRANSLATOR ,I3)
      PRINT 112,(NTHD(L,J),J=1,5)
  112 FORMAT(27H ASSOCIATED THOUSAND DIGITS,23X,5A1/)
      PRINT 113,(NU2H(L,J),J=1,5)
  113 FORMAT(43H ASSOCIATED UNITS OF 200 (1,2,3,4 AND/OR5)
     1,7X,5A1/)
  330 CONTINUE
      GO TO 340
  310 PRINT 114,ME1,ME2
  114 FORMAT(8H ERRORS ,/2I5/)
```

```
340 PRINT 115,NT1,NT2
115 FORMAT(7H NOTES /2I5)
    END
```

```
 ** DECISION TABLE GENERATOR RUN ** ** NON-EXECUTABLE STATEMENTS ** PAGE     1

      PROGRAM TESTER2
      CHARACTER TEMP,NCBLANK,J1,NTHD,NU2H
      DIMENSION STO(8)
      COMMON IOFCO(3),NUMF(3,7),NUMT(3,7),NTCOF(3),NTCOT(3),
     NRCO(3),IVAR(50),TEMP(58),I4,JVAR(50),NTHD(5,5),NCBLANK,
     NU2H(5,5),NBLANK,NTR,ME3,ME4,ME5
  201FORMAT(A6,A3,7A8,A7)
  202FORMAT(1X,A4,58A1)
  203FORMAT(5X,2A4)
  204FORMAT(5X,A2)
  205FORMAT(3A1)
  111FORMAT(12H TRANSLATOR ,I3)
  112FORMAT(27H ASSOCIATED THOUSAND DIGITS,23X,5A1/)
  113FORMAT(43H ASSOCIATED UNITS OF 200 (1,2,3,4 AND/ORS)
  113,7X,5A1/)
  114FORMAT(8H ERRORS ,/2I5/)
  115FORMAT(7H NOTES /2I5)
```

```
I *** TABLE 000.00 --- RELATED TO ORIGINAL LABEL 00000
I CONDITIONS =  0   ACTIONS =  6   RULES =  1
I
I-----------------------------------------------------------------------------
I NBLANK=(4H    )                                                        I
I NCBLANK=(1H )                                                          I X
I KNU=HE1=HE2=NT1=NT2=0                                                  I X
I ASSIGN 10 TO LABEL                                                     I X
I DUMVAR=1                                                               I X
I GO TO 000.01                                                           I X
=============================================================================

I *** TABLE 000.01 --- RELATED TO ORIGINAL LABEL 00000
I CONDITIONS =  5   ACTIONS = 11   RULES =  6
I
I-----------------------------------------------------------------------------
I LABEL.EQ.17                                                            I YYYNNN
I LABEL.EQ.10                                                            I   YNN
I LABEL.EQ.4                                                             I     YN
I DUMVAR.LT.0                                                            I YNN
I DUMVAR.EQ.0                                                            I  YN
I ------------------------------------------------------------------     I ---------
I GO TO 005.00                                                           I    X
I END                                                                    I     X
I DUMVAR=2                                                               I
I GO TO 007.00                                                           I X
I DUMVAR=3                                                               I X
I GO TO 006.00                                                           I  X
I DUMVAR=4                                                               I  X
I I2345=1                                                                I  X X
I J2345= 2                                                               I   X X
I  K2345 = 6                                                             I   X X
I GO TO 005.00                                                           I   X X
=============================================================================

I *** TABLE 005.00 --- RELATED TO ORIGINAL LABEL 00010
I CONDITIONS =  0   ACTIONS =  2   RULES =  1
I
I-----------------------------------------------------------------------------
I READ(60,201)XJOB,ICAR,STO                                              I X
I GO TO 005.01                                                           I X
=============================================================================
```

*** TABLE 005.01 --- RELATED TO ORIGINAL LABEL 00010
CONDITIONS = 3 ACTIONS = 7 RULES = 4

EOFCKF(60).EQ.1	YYNN	
EOFCKF(60).EQ.2	YN	
ME1.EQ.0.AND.ME2.EQ.0	YN	
GO TO 006.00	X	
GO TO 006.00	X	
I1=I3=1	X	
I2=NTR=0	X	
GO TO 022.01	X	
PRINT 114,ME1,ME2	X	
GO TO 042.00	X	

*** TABLE 006.00 --- RELATED TO ORIGINAL LABEL 00020
CONDITIONS = 0 ACTIONS = 12 RULES = 1

KNU=KNU+1	X
	X
**********THE ABOVE ALREADY EXISTS**********	X
	X
	X
READING OF #NUM# CARDS AND INFORMATION STORAGE	X
	X
J1=128	X
DECODE(63,202,STO)IOFCO(KNU),(TEMP(I),I=1,58)	X
I1=0	X
I=1	X
GO TO 006.01	X

*** TABLE 006.01 --- RELATED TO ORIGINAL LABEL 00020
CONDITIONS = 0 ACTIONS = 3 RULES = 1

I3=0	X
J=1	X
GO TO 006.02	X

*** TABLE 006.02 --- RELATED TO ORIGINAL LABEL 00020
CONDITIONS = 0 ACTIONS = 2 RULES = 1

I1=I1+1	X
GO TO 006.03	X

*** TABLE 006.03 --- RELATED TO ORIGINAL LABEL 00020
CONDITIONS = 1 ACTIONS = 2 RULES = 2

TEMP(I1).GE.J1	YN
GO TO 007.00	X
GO TO 009.00	X

*** TABLE 007.00 --- RELATED TO ORIGINAL LABEL 00030
CONDITIONS = 1 ACTIONS = 3 RULES = 2

TEMP(I1).EQ.NCBLANK	YN
GO TO 013.00	X
I3=I3+1	X
GO TO 009.00	X

*** TABLE 009.00 --- RELATED TO ORIGINAL LABEL 00050
CONDITIONS = 0 ACTIONS = 1 RULES = 1

GO TO 009.01	X

*** TABLE 009.01 --- RELATED TO ORIGINAL LABEL 00050
CONDITIONS = 3 ACTIONS = 7 RULES = 4

J=8	YYYN
I3.GT.0.AND.I3.LT.8	YNN
I.EQ.7	YN
J=J+1	X
GO TO 006.02	X
GO TO 013.00	X
DECODE(13,203,STO(I))NUMF(KNU,I),NUMT(KNU,I)	X
GO TO 012.01	X
DECODE(13,203,STO(I))NTCOF(KNU),NTCOT(KNU)	X
GO TO 016.00	X

*** TABLE 012.01 --- RELATED TO ORIGINAL LABEL 00080
CONDITIONS = 1 ACTIONS = 3 RULES = 2

I=7	YN
GO TO 013.00	X
I=I+1	X
GO TO 006.01	X

*** TABLE 013.00 --- RELATED TO ORIGINAL LABEL 00090
CONDITIONS = 1 ACTIONS = 5 RULES = 2

I.EQ.7	YN
NT1=KNU	X
NTCOF(KNU)=NTCOT(KNU)=NBLANK	X
GO TO 016.00	X
ME1=KNU	X
GO TO 005.00	X

*** TABLE 016.00 --- RELATED TO ORIGINAL LABEL 00120
CONDITIONS = 2 ACTIONS = 4 RULES = 3

TEMP(57).EQ.NCBLANK.AND.TEMP(58).EQ.NCBLANK	YNN
TEMP(57).LT.J1.AND.TEMP(58).LT.J1	YN
DECODE(7,204,STO(81))NRCO(KNU)	XX
GO TO 005.00	XX
ME2=KNU	X
GO TO 005.00	X

*** TABLE 022.01 --- RELATED TO ORIGINAL LABEL 00300
CONDITIONS = 1 ACTIONS = 3 RULES = 2

KNU.EQ.1	YN
GO TO 038.00	X
J=1	X
GO TO 023.01	X

```
*** TABLE 023.01 --- RELATED TO ORIGINAL LABEL 00170
CONDITIONS =  0   ACTIONS =  3   RULES =  1

DECODE(3,205,IOFCO(J))(TEMP(I),I=1,3)
I=1                                                    X
GO TO 023.02                                           X
                                                       X

*** TABLE 023.02 --- RELATED TO ORIGINAL LABEL 00170
CONDITIONS =  6   ACTIONS = 12   RULES =  7

TEMP(I).EQ.NCBLANK                                     YNNNNNN
I=3                                                     YYYYYN
I2.EQ.1                                                 YYYYN
IOFCO(J).EQ.IOFCO(J-1)                                  YNNN
(J+1).GT.KNU                                             YNN
IOFCO(J+1).EQ.IOFCO(1)                                    YN
---------------------------------------------------------------
I=I+1                                                        X
GO TO 023.02                                                 X
GO TO 036.00                                            X
CALL TRASU(I1,J-1,I3)                                    X X
GO TO 028.01                                             X X
I1=I3=2                                                   X
I2=J                                                         X
GO TO 032.00                                                 X
NT2=1                                                   X
GO TO 034.01                                            X
CALL TRASU(1,3,2)                                          X
GO TO 037.01                                               X

*** TABLE 028.01 --- RELATED TO ORIGINAL LABEL 00200
CONDITIONS =  1   ACTIONS =  3   RULES =  2

ME3.NE.0.OR.ME4.NE.0.OR.ME5.NE.0                       YN
---------------------------------------------------------------
GO TO 039.00                                           X
I1=J                                                    X
GO TO 036.00
```

```
I *** TABLE 032.00 --- RELATED TO ORIGINAL LABEL 00240
I CONDITIONS =  1   ACTIONS =  4   RULES =  2
I
I-------------------------------------------------------------------------------
I I2.EQ.1                                                          I YN
I ------------------------------------------------------------------ I ----------
I GO TO 036.00                                                     I X
I I2=I2-1                                                          I  X
I IOFCO(I2)=IOFCO(J)                                               I  X
I GO TO 032.00                                                     I  X
```

```
I *** TABLE 034.01 --- RELATED TO ORIGINAL LABEL 00260
I CONDITIONS =  1   ACTIONS =  3   RULES =  2
I
I-------------------------------------------------------------------------------
I I2.EQ.1                                                          I YN
I ------------------------------------------------------------------ I ----------
I GO TO 036.00                                                     I  X
I IOFCO(J)=IOFCO(J-1)                                              I X
I GO TO 036.00                                                     I X
```

```
I *** TABLE 036.00 --- RELATED TO ORIGINAL LABEL 00280
I CONDITIONS =  0   ACTIONS =  1   RULES =  1
I
I-------------------------------------------------------------------------------
I GO TO 036.01                                                     I X
```

```
I *** TABLE 036.01 --- RELATED TO ORIGINAL LABEL 00280
I CONDITIONS =  1   ACTIONS =  3   RULES =  2
I
I-------------------------------------------------------------------------------
I J=KNU                                                            I YN
I ------------------------------------------------------------------ I ----------
I J=J+1                                                            I  X
I GO TO 023.01                                                     I  X
I GO TO 038.00                                                     I X
```

*** TABLE 037.01 --- RELATED TO ORIGINAL LABEL 00320
CONDITIONS = 1 ACTIONS = 2 RULES = 2

ME3.NE.0.OR.ME4.NE.0.OR.ME5.NE.0	YN	
GO TO 039.00	X	
GO TO 038.00		X

*** TABLE 038.00 --- RELATED TO ORIGINAL LABEL 00290
CONDITIONS = 0 ACTIONS = 2 RULES = 1

CALL TRASU(I1,KNU,I3)	X
GO TO 039.00	X

*** TABLE 039.00 --- RELATED TO ORIGINAL LABEL 00860
CONDITIONS = 0 ACTIONS = 2 RULES = 1

PRINTING

L=1	X
GO TO 039.01	X

*** TABLE 039.01 --- RELATED TO ORIGINAL LABEL 00860
CONDITIONS = 0 ACTIONS = 5 RULES = 1

J2=L-1	X
PRINT 111,J2	X
PRINT 112,(NTHD(L,J),J=1,5)	X
PRINT 113,(NU2H(L,J),J=1,5)	X
GO TO 040.01	X

```
I *** TABLE 040.01 --- RELATED TO ORIGINAL LABEL 00330                                  I
I CONDITIONS =  1   ACTIONS =  3   RULES =  2                                          I
I                                                                                      I
I--------------------------------------------------------------------------------------I
I L=NTR                                                          I YN                  I
I ---------------------------------------------------------------I ------------------- I
I L=L+1                                                          I  X                  I
I GO TO 039.01                                                   I  X                  I
I GO TO 042.00                                                   I X                   I
```

```
I *** TABLE 042.00 --- RELATED TO ORIGINAL LABEL 00340                                  I
I CONDITIONS =  0   ACTIONS =  2   RULES =  1                                          I
I                                                                                      I
I--------------------------------------------------------------------------------------I
I PRINT 115,NT1,NT2                                              I X                   I
I END                                                            I X                   I
```

```
00.00  REFERS TO      000.01
       REFERENCED BY FIRST

00.01  REFERS TO      005.00  007.00  006.00
       REFERENCED BY 000.00

05.00  REFERS TO      005.01
       REFERENCED BY 000.01  016.00  013.00

05.01  REFERS TO      006.00  022.01  042.00
       REFERENCED BY 005.00

06.00  REFERS TO      006.01
       REFERENCED BY 000.01  005.01

06.01  REFERS TO      006.02
       REFERENCED BY 006.00  012.01

06.02  REFERS TO      006.03
       REFERENCED BY 006.01  009.01

06.03  REFERS TO      007.00  009.00
       REFERENCED BY 006.02

07.00  REFERS TO      013.00  009.00
       REFERENCED BY 000.01  006.03

09.00  REFERS TO      009.01
       REFERENCED BY 006.03  007.00

09.01  REFERS TO      006.02  013.00  012.01  016.00
       REFERENCED BY 009.00

12.01  REFERS TO      013.00  006.01
       REFERENCED BY 009.01

13.00  REFERS TO      016.00  005.00
       REFERENCED BY 007.00  009.01  012.01

16.00  REFERS TO      005.00
       REFERENCED BY 013.00  009.01

22.01  REFERS TO      038.00  023.01
       REFERENCED BY 005.01

23.01  REFERS TO      023.02
       REFERENCED BY 022.01  036.01

23.02  REFERS TO      023.02  036.00  028.01  032.00  034.01  037.01
       REFERENCED BY 023.01  023.02

28.01  REFERS TO      039.00  036.00
       REFERENCED BY 023.02

32.00  REFERS TO      036.00  032.00
       REFERENCED BY 023.02  032.00
```

```
034.01  REFERS TO      036.00
        REFERENCED BY  023.02

036.00  REFERS TO      036.01
        REFERENCED BY  023.02  028.01  032.00  034.01

036.01  REFERS TO      023.01  038.00
        REFERENCED BY  036.00

037.01  REFERS TO      039.00  038.00
        REFERENCED BY  023.02

038.00  REFERS TO      039.00
        REFERENCED BY  022.01  036.01  037.01

039.00  REFERS TO      039.01
        REFERENCED BY  028.01  037.01  038.00

039.01  REFERS TO      040.01
        REFERENCED BY  039.00  040.01

040.01  REFERS TO      039.01  042.00
        REFERENCED BY  039.01

042.00  REFERS TO      NO TABLES *****
        REFERENCED BY  040.01  005.01
```

APPENDIX C

DIFFERENT FORMS OF THE 'IF' STATEMENT

IF ACCUMULATOR OVERFLOW n1,n2

IF QUOTIENT OVERFLOW n1,n2

IF (arithmetic expression) n1,n2,n3

IF (logical expression) n1,n2

IF (logical expression) statement

IF DIVIDE CHECK n1,n2

IF (ENDFILE i) n1,n2

IF (EOF,i) n1,n2

IF (SENSE LIGHT i) n1,n2

IF (SENSE SWITCH i) n1,n2

IF (UNIT,i) n1,n2,n3,n4

APPENDIX D

PROGRAM LISTINGS

```
0001    000100 IDENTIFICATION DIVISION.
0002    000200 PROGRAM-ID. DETBL00.
0003    000300 AUTHOR. V HOLLMANN.
0004    000400 INSTALLATION. SIR GEORGE WILLIAMS UNIVERSITY.
0005    000500 DATE-COMPILED. JANUARY  75/07/18..
0006    000600 REMARKS.
0007    000700          THIS IS THE CONTROL PROGRAM FOR THE DECISION TABLE
0008    000800      GENERATING SYSTEM.
0009    000900 ENVIRONMENT DIVISION.
0010    001000 CONFIGURATION SECTION.
0011    001100 SOURCE-COMPUTER. 6400.
0012    001200 OBJECT-COMPUTER. 6400.
0013    001300 INPUT-OUTPUT SECTION.
0014    001400 FILE-CONTROL.
0015    001500      SELECT KONTROL ASSIGN TO KONTROL.
0016    001600      SELECT INTER1 ASSIGN TO INTER1.
0017    001700      SELECT INTER2 ASSIGN TO INTER2.
0018    001800      SELECT SOURCE-DECK ASSIGN TO SOURCIN.
0019    001900      SELECT LNEFILE ASSIGN TO LNEFILE.
0020    002000      SELECT PRTFILE ASSIGN TO PRTFILE.
0021    002100      SELECT DTINDEX ASSIGN TO DTINDEX
0022    002200          ORGANIZATION IS STANDARD
0023    002300          FILE-LIMIT IS 1000
0024    002400          ACCESS MODE IS RANDOM
0025    002500          SYMBOLIC KEY IS REC-KEY.
0026    002600      SELECT DETABLS ASSIGN TO DECTBLS
0027    002700          FILE-LIMIT IS 2000
0028    002800          ACCESS IS RANDOM
0029    002900          ACTUAL KEY IS TBL-KEY.
0030    003000
0031    003100 DATA DIVISION.
0032    003200 FILE SECTION.
0033    003300 FD  KONTROL
0034    003400      LABEL RECORDS ARE OMITTED
0035    003500      BLOCK CONTAINS 1 RECORDS
0036    003600      RECORD CONTAINS 80 CHARACTERS
0037    003700      DATA RECORD IS CTRL-REC.
0038    003800
0039    003900 01  CTRL-REC              PIC X(80).
0040    004000
0041    004100 FD  INTER1
0042    004200      LABEL RECORDS ARE OMITTED
0043    004300      BLOCK CONTAINS 20 RECORDS
0044    004400      RECORD CONTAINS 76 CHARACTERS
0045    004500      DATA RECORD IS INTCARD.
0046    004600
0047    004700 01  INTCARD                PIC X(76).
0048    004800
0049    004900 FD  INTER2
0050    005000      LABEL RECORDS ARE OMITTED
0051    005100      BLOCK CONTAINS 20 RECORDS
0052    005200      RECORD CONTAINS 80 CHARACTERS
0053    005300      DATA RECORD IS INTREC2.
0054    005400
0055    005500 01  INTREC2                PIC X(80).
0056    005600
```

```
0057     005700 FD  SOURCE-DECK
0058     005800     LABEL RECORDS ARE OMITTED
0059     005900     BLOCK CONTAINS 1 RECORDS
0060     006000     RECORD CONTAINS 80 CHARACTERS
0061     006100     DATA RECORD IS CARD.
0062     006200
0063     006300 01  CARD                    PIC X(80).
0064     006400
0065     006500 FD  LNEFILE
0066     006600     LABEL RECORDS ARE OMITTED
0067     006700     BLOCK CONTAINS 20 RECORDS
0068     006800     RECORD CONTAINS 133 CHARACTERS
0069     006900     DATA RECORD IS PRNT-RECD.
0070     007000
0071     007100 01  PRNT-RECD               PIC X(133).
0072     007200
0073     007300 FD  PRTFILE
0074     007400     LABEL RECORDS ARE OMITTED
0075     007500     BLOCK CONTAINS 20 RECORDS
0076     007600     RECORD CONTAINS 133 CHARACTERS
0077     007700     DATA RECORD IS PRNT-RECD.
0078     007800
0079     007900 01  PRNT-RECD               PIC X(133).
0080     008000
0081     008100 FD  DTINDEX
0082     008200     BLOCK CONTAINS 1 RECORDS
0083     008300     RECORD CONTAINS 60 CHARACTERS
0084     008400     DATA RECORD IS DTBL-PNTR
0085     008500     LABEL RECORDS ARE OMITTED.
0086     008600
0087     008700 01  DTBL-PNTR               PIC X(60).
0088     008800
0089     008900 FD  DETABLS
0090     009000     LABEL RECORDS ARE OMITTED
0091     009100     BLOCK CONTAINS 1 RECORDS
0092     009200     RECORD CONTAINS 132 CHARACTERS
0093     009300     DATA RECORD IS DTBL-NTRY.
0094     009400
0095     009500 01  DTBL-NTRY               PIC X(132).
0096     009600
0097     009700 COMMON-STORAGE SECTION.
0098     009800
0099     009900 01  COMN-INFO.
0100     010000     05  RETN-CODE    PIC S9(4)  COMP VALUE 0.
0101     010100     05  PASD-INFO    PIC X(138)      VALUE SPACE.
0102     010200
0103     010300 WORKING-STORAGE SECTION.
0104     010400 77  TBL-KEY                 PIC 9(5)        VALUE 0.
0105     010500 77  REC-KEY                 PIC X(5)        VALUE LOW-VALUES.
0106     010600 77  EROR-INDX               PIC S99         VALUE +2 -      COMP.
0107     010700 77  EROR-FLAG               PIC 9           VALUE 0.
0108     010800 77  INDX-GP01               PIC 99          VALUE 0 COMP.
0109     010900 77  INDX-GP02               PIC 99          VALUE 0 COMP.
0110     011000
0111     011100 01  CTRL-AREA.
0112     011200     05  CARD-IDNT           PIC XXX         VALUE SPACE.
```

```
011300     05  CARD-CHAR       PIC X          OCCURS 73 TIMES.
011400
011500 01  REPT-FLGS.
011600     05  FLAG-AREA.
011700         10  REPT-FLAG   PIC X          OCCURS 4 TIMES.
011800     05  REPT-CODS       PIC X(4)       VALUE ≠CEFH≠.
011900     05  REPT-CODE REDEFINES REPT-CODS PIC X OCCURS 4 TIMES.
012000     05  LIST-FLGS.
012100         10  C-FLAG      PIC X          VALUE SPACE.
012200         10  E-FLAG      PIC X          VALUE SPACE.
012300         10  F-FLAG      PIC X          VALUE SPACE.
012400         10  H-FLAG      PIC X          VALUE SPACE.
012500
012600 01  CARD-INFO.
012700     05  INFO-AREA.
012800         10  INFO-CHAR   PIC X          OCCURS 73 TIMES.
012900     05  LANG-AREA REDEFINES INFO-AREA.
013000         10  CREA-CNST    PIC X(7).
013100         10  LANG-NAME   PIC X(8).
013200         10  LANG-FL05   PIC X(58).
013300     05  OPTM-AREA REDEFINES INFO-AREA.
013400         10  OPTM-CNST   PIC X(6).
013500         10  OPTM-PASS.
013600             15  OPTM-PASN   PIC 9.
013700         10  OPTM-FL05   PIC X(66).
013800     05  PRNT-AREA REDEFINES INFO-AREA.
013900         10  PRNT-CNST   PIC X(6).
014000         10  PRNT-FL05.
014100             15  PRNT-RPTS   PIC XXX.
014200             15  PRNT-FL10   PIC X(64).
014300         10  PRNT-CHAR REDEFINES PRNT-FL05 PIC X OCCURS 67 TIMES.
014400     05  LIST-AREA REDEFINES INFO-AREA.
014500         10  LIST-CNST   PIC X(5).
014600         10  LIST-FL05.
014700             15  LIST-RPTS   PIC XXX.
014800             15  LIST-FL10   PIC X(65).
014900         10  LIST-CHAR REDEFINES LIST-FL05 PIC X OCCURS 68 TIMES.
015000
015100 01  MESG-LINE.
015200     05  MESG-IDNT       PIC X(9)       VALUE ≠*** DET00≠.
015300     05  MESG-NMBR       PIC 99         VALUE 0.
015400     05  MESG-FL05       PIC X(5)       VALUE ≠ --- ≠.
015500     05  MESG-AREA       PIC X(31)      VALUE SPACE.
015600     05  EROR-VALU       PIC X(86)      VALUE SPACE.
015700
015800 01  EROR-MSGS.
015900     05  MESG-VALU.
016000         10  MESG-01 PIC X(30) VALUE ≠CONTROL CARD(S) MISSING      ≠.
016100         10  MESG-02 PIC X(30) VALUE ≠JOB TERMINATED SUCCESSFULLY  ≠.
016200         10  MESG-03 PIC X(30) VALUE ≠RUN DELETED                  ≠.
016300         10  MESG-04 PIC X(30) VALUE ≠LANGUAGE NOT SPECIFIED       ≠.
016400         10  MESG-05 PIC X(30) VALUE ≠TABLE MERGE SPECIFIED        ≠.
016500         10  MESG-06 PIC X(30) VALUE ≠LANGUAGE NOT SUPPORTED       ≠.
016600         10  MESG-07 PIC X(30) VALUE ≠INTERPRETATION UNSUCCESSFUL  ≠.
016700         10  MESG-08 PIC X(30) VALUE ≠UNABLE TO CREATE TABLES      ≠.
016800         10  MESG-09 PIC X(30) VALUE ≠ERROR DURING TABLE MERGE     ≠.
```

```
169      016900          10  MESG-10 PIC X(30) VALUE ≠UNABLE TO PRINT TABLES        ≠.
170      017000          10  MESG-11 PIC X(30) VALUE ≠INVALID CONTROL CARD          ≠.
171      017100          10  MESG-12 PIC X(30) VALUE ≠INVALID CONTROL CARD FORMAT   ≠.
172      017200          10  MESG-13 PIC X(30) VALUE ≠TABLE MERGE PASSES NON NUMERIC≠.
173      017300          10  MESG-14 PIC X(30) VALUE ≠TOO MANY TABLE MERGE PASSES   ≠.
174      017400          10  MESG-15 PIC X(30) VALUE ≠INVALID REPORT REQUEST USE ALL≠.
175      017500          05  EROR-MESG REDEFINES MESG-VALU PIC X(30) OCCURS 15 TIMES.
176      017600
177      017700  01  LIST-CARD.
178      017800          05  LIST-ITEM OCCURS 4 TIMES.
179      017900              10  LIST-REPT   PIC X(4).
180      018000              10  LIST-FL15   PIC X.
181      018100
182      018200  01  LIST-REDF REDEFINES LIST-CARD.
183      018300          05  LIST-BYTE       PIC X          OCCURS 20 TIMES.
184      018400
```

```
018500 PROCEDURE DIVISION.
018600 000-PROGRAM-START SECTION 1.
018700 000-START.
018800     OPEN INPUT KONTROL.
018900     MOVE SPACE TO FLAG-AREA.
019000     READ KONTROL INTO CTRL-AREA AT END
019100         MOVE +1 TO EROR-INDX
019200         PERFORM 900-WRITE-MESG THRU 909-WRITE-MESG-END
019300         MOVE +3 TO EROR-INDX
019400         GO TO 999-ENDIT.
019500     CALL ≠DETBL90≠ USING COMN-INFO.
019600
019700 050-PROCESS.
019800     PERFORM 700-INTERPRET THRU 799-INTERPRET-END.
019900     PERFORM 800-EDIT-CONTROL THRU 899-EDIT-CONTROL-END.
020000     IF EROR-FLAG = ≠2≠
020100         MOVE +3 TO EROR-INDX
020200         GO TO 999-ENDIT.
020300     IF EROR-FLAG = ≠1≠
020400         MOVE ≠0≠ TO EROR-FLAG
020500         GO TO 070-READ.
020600     IF CREA-CNST = ≠CREATE=≠
020700         PERFORM 100-LANGUAGE THRU 199-LANGUAGE-END
020800         PERFORM 200-CREATE THRU 299-CREATE-END
020900         GO TO 070-READ.
021000     IF OPTM-CNST = ≠MERGE=≠
021100         MOVE OPTM-PASS TO PASD-INFO
021200         PERFORM 300-OPTIMIZE THRU 399-OPTIMIZE-END
021300         GO TO 070-READ.
021400     IF PRNT-CNST = ≠PRINT=≠
021500         MOVE PRNT-RPTS TO PASD-INFO
021600         PERFORM 400-PRINT THRU 499-PRINT-END
021700         GO TO 070-READ.
021800     IF LIST-CNST = ≠LIST=≠
021900         PERFORM 500-CHOOSE-REPORTS THRU 599-CHOOSE-REPORTS-END
022000         GO TO 070-READ.
022100
022200 070-READ.
022300     READ KONTROL INTO CTRL-AREA AT END
022400         GO TO 910-TERMINATE.
022500     GO TO 050-PROCESS.
022600
022700 100-LANGUAGE.
022800     IF LANG-NAME = ≠FORTRAN ≠
022900         MOVE FLAG-AREA TO PASD-INFO
023000         CALL ≠DETBL40≠ USING COMN-INFO
023100         GO TO 150-CHECK-RETN.
023200
023300 150-CHECK-RETN.
023400     IF RETN-CODE NOT = 0
023500         MOVE +7 TO EROR-INDX
023600         PERFORM 900-WRITE-MESG THRU 909-WRITE-MESG-END
023700         MOVE +3 TO EROR-INDX
023800         GO TO 910-TERMINATE.
023900
024000 199-LANGUAGE-END.
```

```
241      024100     EXIT.
242      024200
243      024300 200-CREATE.
244      024400*    CALL ≠DETBL10≠ USING COMN-INFO.
245      024500     IF RETN-CODE NOT = 0
246      024600         MOVE +8 TO EROR-INDX
247      024700         PERFORM 900-WRITE-MESG THRU 909-WRITE-MESG-END
248      024800         MOVE +3 TO EROR-INDX
249      024900         GO TO 910-TERMINATE.
250      025000
251      025100 299-CREATE-END.
252      025200     EXIT.
253      025300
254      025400 300-OPTIMIZE.
255      025500     CALL ≠DETBL20≠ USING COMN-INFO.
256      025600     IF RETN-CODE NOT = 0
257      025700         MOVE +9 TO EROR-INDX
258      025800         PERFORM 900-WRITE-MESG THRU 909-WRITE-MESG-END
259      025900         MOVE +3 TO EROR-INDX
260      026000         GO TO 910-TERMINATE.
261      026100
262      026200 399-OPTIMIZE-END.
263      026300     EXIT.
264      026400
265      026500 400-PRINT.
266      026600     IF PRNT-RPTS = ≠ALL≠
267      026700         MOVE ≠FH≠ TO PRNT-RPTS
268      026800         GO TO 450-CALL-PRINT.
269      026900     MOVE +0 TO INDX-GP01.
270      027000     MOVE SPACE TO PRNT-RPTS.
271      027100     IF F-FLAG NOT EQUAL SPACE
272      027200         ADD +1 TO INDX-GP01
273      027300         MOVE ≠F≠ TO PRNT-CHAR (INDX-GP01).
274      027400     IF H-FLAG NOT EQUAL SPACE
275      027500         ADD +1 TO INDX-GP01
276      027600         MOVE ≠H≠ TO PRNT-CHAR (INDX-GP01).
277      027700
278      027800 450-CALL-PRINT.
279      027900     MOVE PRNT-RPTS TO PASD-INFO.
280      028000     CALL ≠DETBL30≠ USING COMN-INFO.
281      028100     IF RETN-CODE = 0
282      028200         GO TO 499-PRINT-END.
283      028300     MOVE +10 TO EROR-INDX
284      028400     PERFORM 900-WRITE-MESG THRU 909-WRITE-MESG-END.
285      028500     MOVE +3 TO EROR-INDX.
286      028600     GO TO 910-TERMINATE.
287      028700
288      028800 499-PRINT-END.
289      028900     EXIT.
290      029000
291      029100 500-CHOOSE-REPORTS.
292      029200     MOVE +0 TO INDX-GP01.
293      029300     IF LIST-RPTS = ≠ALL≠
294      029400         MOVE REPT-CODS TO FLAG-AREA
295      029500         GO TO 599-CHOOSE-REPORTS-END.
296      029600     IF C-FLAG NOT EQUAL SPACE
```

```
297      029700          ADD +1 TO INDX-GP01
298      029800          MOVE ≠C≠ TO REPT-FLAG (INDX-GP01).
299      029900      IF E-FLAG NOT EQUAL SPACE
300      030000          ADD +1 TO INDX-GP01
301      030100          MOVE ≠E≠ TO REPT-FLAG (INDX-GP01).
302      030200      IF F-FLAG NOT EQUAL SPACE
303      030300          ADD +1 TO INDX-GP01
304      030400          MOVE ≠F≠ TO REPT-FLAG (INDX-GP01).
305      030500      IF H-FLAG NOT EQUAL SPACE
306      030600          ADD +1 TO INDX-GP01
307      030700          MOVE ≠H≠ TO REPT-FLAG (INDX-GP01).
308      030800
309      030900 599-CHOOSE-REPORTS-END.
310      031000      EXIT.
311      031100
312      031200 700-INTERPRET.
313      031300      MOVE 0 TO INDX-GP01.
314      031400      MOVE 0 TO INDX-GP02.
315      031500      MOVE SPACE TO CARD-INFO.
316      031600
317      031700 710-MOVE.
318      031800      ADD 1 TO INDX-GP01.
319      031900      IF INDX-GP01 IS GREATER THAN 73
320      032000          GO TO 799-INTERPRET-END.
321      032100      IF CARD-CHAR (INDX-GP01) = SPACE
322      032200          GO TO 710-MOVE.
323      032300      ADD 1 TO INDX-GP02.
324      032400      MOVE CARD-CHAR (INDX-GP01) TO INFO-CHAR (INDX-GP02).
325      032500      GO TO 710-MOVE.
326      032600
327      032700 799-INTERPRET-END.
328      032800      EXIT.
329      032900
330      033000 800-EDIT-CONTROL.
331      033100      IF CARD-IDNT NOT = ≠****≠
332      033200          MOVE +12 TO EROR-INDX
333      033300          MOVE CARD-IDNT TO EROR-VALU
334      033400          PERFORM 900-WRITE-MESG THRU 909-WRITE-MESG-END
335      033500          MOVE ≠2≠ TO EROR-FLAG
336      033600          GO TO 899-EDIT-CONTROL-END.
337      033700      IF CREA-CNST = ≠CREATE=≠
338      033800          GO TO 810-EDIT-CREATE.
339      033900      IF OPTM-CNST = ≠MERGE=≠
340      034000          GO TO 830-EDIT-OPTIMIZE.
341      034100      IF PRNT-CNST = ≠PRINT=≠
342      034200          GO TO 850-EDIT-PRINT.
343      034300      IF LIST-CNST = ≠LIST=≠
344      034400          GO TO 870-EDIT-LIST.
345      034500      MOVE +11 TO EROR-INDX.
346      034600      MOVE CARD-INFO TO EROR-VALU.
347      034700      PERFORM 900-WRITE-MESG THRU 909-WRITE-MESG-END.
348      034800      MOVE ≠2≠ TO EROR-FLAG.
349      034900      GO TO 899-EDIT-CONTROL-END.
350      035000
351      035100 810-EDIT-CREATE.
352      035200      IF LANG-NAME = ≠FORTRAN ≠
```

```
353     035300          GO TO 899-EDIT-CONTROL-END.
354     035400      IF LANG-NAME = SPACE
355     035500          MOVE +4 TO EROR-INDX
356     035600          PERFORM 900-WRITE-MESG THRU 909-WRITE-MESG-END
357     035700          MOVE ≠2≠ TO EROR-FLAG.
358     035800      MOVE +6 TO EROR-INDX.
359     035900      MOVE LANG-NAME TO EROR-VALU.
360     036000      PERFORM 900-WRITE-MESG THRU 909-WRITE-MESG-END.
361     036100      MOVE ≠2≠ TO EROR-FLAG.
362     036200      GO TO 899-EDIT-CONTROL-END.
363     036300
364     036400 830-EDIT-OPTIMIZE.
365     036500      IF OPTM-PASS NOT NUMERIC
366     036600          MOVE OPTM-PASS TO EROR-VALU
367     036700          MOVE +13 TO EROR-INDX
368     036800          PERFORM 900-WRITE-MESG THRU 909-WRITE-MESG-END
369     036900          MOVE 1 TO OPTM-PASN
370     037000          GO TO 899-EDIT-CONTROL-END.
371     037100      IF OPTM-PASN IS GREATER THAN 5
372     037200          MOVE +14 TO EROR-INDX
373     037300          MOVE OPTM-PASS TO EROR-VALU
374     037400          PERFORM 900-WRITE-MESG THRU 909-WRITE-MESG-END
375     037500          MOVE 1 TO OPTM-PASN.
376     037600      GO TO 899-EDIT-CONTROL-END.
377     037700
378     037800 850-EDIT-PRINT.
379     037900      IF PRNT-RPTS = ≠ALL≠
380     038000          GO TO 899-EDIT-CONTROL-END.
381     038100      MOVE SPACE TO LIST-CARD.
382     038200      MOVE +0 TO INDX-GP01.
383     038300      MOVE +0 TO INDX-GP02.
384     038400
385     038500 855-MOVE-CHARS.
386     038600      ADD +1 TO INDX-GP01.
387     038700      IF INDX-GP01 IS GREATER THAN +67
388     038800          GO TO 860-EDIT-CODES.
389     038900      IF PRNT-CHAR (INDX-GP01) = SPACE
390     039000          GO TO 855-MOVE-CHARS.
391     039100      ADD +1 TO INDX-GP02.
392     039200      IF INDX-GP02 IS GREATER THAN +10
393     039300          GO TO 869-PRNT-ERROR.
394     039400      MOVE PRNT-CHAR (INDX-GP01) TO LIST-BYTE (INDX-GP02).
395     039500      GO TO 855-MOVE-CHARS.
396     039600
397     039700 860-EDIT-CODES.
398     039800      MOVE SPACE TO F-FLAG.
399     039900      MOVE +0 TO INDX-GP01.
400     040000      MOVE SPACE TO H-FLAG.
401     040100
402     040200 865-EDIT-CODES.
403     040300      ADD +1 TO INDX-GP01.
404     040400      IF LIST-REPT (INDX-GP01) = ≠TRLS≠
405     040500          MOVE ≠X≠ TO F-FLAG
406     040600          GO TO 865-EDIT-CODES.
407     040700      IF LIST-REPT (INDX-GP01) = ≠XREF≠
408     040800          MOVE ≠X≠ TO H-FLAG
```

```
409      040900          GO TO 865-EDIT-CODES.
410      041000      IF LIST-REPT (INDX-GP01) = SPACE
411      041100          GO TO 899-EDIT-CONTROL-END.
412      041200
413      041300 869-PRNT-ERROR.
414      041400      MOVE ≠ALL≠ TO PRNT-RPTS.
415      041500      MOVE LIST-REPT (INDX-GP01) TO EROR-VALU.
416      041600      MOVE +15 TO EROR-INDX.
417      041700      PERFORM 900-WRITE-MESG THRU 909-WRITE-MESG-END.
418      041800      GO TO 899-EDIT-CONTROL-END.
419      041900
420      042000 870-EDIT-LIST.
421      042100      IF LIST-RPTS = ≠ALL≠
422      042200          GO TO 899-EDIT-CONTROL-END.
423      042300      MOVE SPACE TO LIST-CARD.
424      042400      MOVE +0 TO INDX-GP01.
425      042500      MOVE +0 TO INDX-GP02.
426      042600
427      042700 875-MOVE-CHARS.
428      042800      ADD +1 TO INDX-GP01.
429      042900      IF INDX-GP01 IS GREATER THAN +68
430      043000          GO TO 880-EDIT-CODES.
431      043100      IF LIST-CHAR (INDX-GP01) = SPACE
432      043200          GO TO 875-MOVE-CHARS.
433      043300      ADD +1 TO INDX-GP02.
434      043400      IF INDX-GP02 IS GREATER THAN +20
435      043500          GO TO 895-LIST-ERROR.
436      043600      MOVE LIST-CHAR (INDX-GP01) TO LIST-BYTE (INDX-GP02).
437      043700      GO TO 875-MOVE-CHARS.
438      043800
439      043900 880-EDIT-CODES.
440      044000      MOVE +0 TO INDX-GP01.
441      044100
442      044200 882-EDIT-CODES.
443      044300      ADD +1 TO INDX-GP01.
444      044400      IF INDX-GP01 IS GREATER THAN +4
445      044500          GO TO 899-EDIT-CONTROL-END.
446      044600      IF LIST-REPT (INDX-GP01) = ≠SORC≠
447      044700          MOVE ≠X≠ TO C-FLAG
448      044800          GO TO 882-EDIT-CODES.
449      044900      IF LIST-REPT (INDX-GP01) = ≠NONX≠
450      045000          MOVE ≠X≠ TO E-FLAG
451      045100          GO TO 882-EDIT-CODES.
452      045200      IF LIST-REPT (INDX-GP01) = ≠TRLS≠
453      045300          MOVE ≠X≠ TO F-FLAG
454      045400          GO TO 882-EDIT-CODES.
455      045500      IF LIST-REPT (INDX-GP01) = ≠XREF≠
456      045600          MOVE ≠X≠ TO H-FLAG
457      045700          GO TO 882-EDIT-CODES.
458      045800      IF LIST-REPT (INDX-GP01) = SPACE
459      045900          GO TO 899-EDIT-CONTROL-END.
460      046000
461      046100 895-LIST-ERROR.
462      046200      MOVE ≠ALL≠ TO LIST-RPTS.
463      046300      MOVE +15 TO EROR-INDX.
464      046400      MOVE LIST-REPT (INDX-GP01) TO EROR-VALU.
```

```
 465      046500      PERFORM 900-WRITE-MESG THRU 909-WRITE-MESG-END.
 466      046600      GO TO 899-EDIT-CONTROL-END.
 467      046700
 468      046800  899-EDIT-CONTROL-END.
 469      046900      EXIT.
 470      047000
 471      047100  900-WRITE-MESG.
 472      047200      MOVE EROR-MESG (EROR-INDX) TO MESG-AREA.
 473      047300      MOVE EROR-INDX TO MESG-NMBR.
 474      047400      DISPLAY MESG-LINE.
 475      047500      MOVE +2 TO EROR-INDX.
 476      047600      MOVE SPACE TO MESG-AREA.
 477      047700      MOVE SPACE TO EROR-VALU.
 478      047800
 479      047900  909-WRITE-MESG-END.
 480      048000      EXIT.
 481      048100
 482      048200  910-TERMINATE.
 483      048300      MOVE FLAG-AREA TO PASD-INFO.
 484      048400      CALL ≠DETBL92≠ USING COMN-INFO.
 485      048500
 486      048600  999-ENDIT.
 487      048700      PERFORM 900-WRITE-MESG THRU 909-WRITE-MESG-END.
 488      048800      CLOSE KONTROL.
 489      048900      STOP RUN.
00 LENGTH IS  002112
00P SCM USED
```

```
001    000100 IDENTIFICATION DIVISION.
002    000200 PROGRAM-ID. DETBL10.
003    000300 INSTALLATION. SIR GEORGE WILLIAMS UNIVERSITY.
004    000400 DATE-COMPILED. JUNE 20, 75/07/18..
005    000500 ENVIRONMENT DIVISION.
006    000600 CONFIGURATION SECTION.
007    000700 SOURCE-COMPUTER. 6400.
008    000800 OBJECT-COMPUTER. 6400.
009    000900 INPUT-OUTPUT SECTION.
010    001000 FILE-CONTROL.
011    001100     SELECT FRMTDFLE ASSIGN TO INTER2.
012    001200     SELECT DTINDEX ASSIGN TO DTINDEX
013    001300         ORGANIZATION IS STANDARD
014    001400         FILE-LIMIT IS 1000
015    001500         ACCESS MODE IS RANDOM
016    001600         SYMBOLIC KEY IS REC-KEY.
017    001700     SELECT DETABLS ASSIGN TO DECTBLS
018    001800         FILE-LIMIT IS 2000
019    001900         ACCESS IS RANDOM
020    002000         ACTUAL KEY IS TBL-KEY.
021    002100
022    002200 DATA DIVISION.
023    002300 FILE SECTION.
024    002400 FD  FRMTDFLE
025    002500     BLOCK CONTAINS 20 RECORDS
026    002600     RECORD CONTAINS 80 CHARACTERS
027    002700     DATA RECORD IS SEQN-RECD
028    002800     LABEL RECORDS ARE OMITTED.
029    002900
030    003000 01  SEQN-RECD.
031    003100     03  RECD-CODE        PIC X.
032    003200     03  RECD-BODY.
033    003300         05  RECD-GOTO    PIC X(6).
034    003400         05  RECD-NAME    PIC X(5).
035    003500         05  RECD-FL01    PIC X(68).
036    003600
037    003700 FD  DTINDEX
038    003800     BLOCK CONTAINS 1 RECORDS
039    003900     RECORD CONTAINS 60 CHARACTERS
040    004000     DATA RECORD IS DTBL-PNTR
041    004100     LABEL RECORDS ARE OMITTED.
042    004200
043    004300 01  DTBL-PNTR.
044    004400     05  TABL-NMBR        PIC X(5).
045    004500     05  TABL-NEXT        PIC X(5).
046    004600     05  TABL-LAST        PIC X(5).
047    004700     05  TABL-INDX        PIC 9(5).
048    004800     05  TABL-RFTO        PIC 9(5).
049    004900     05  TABL-RFBY        PIC 9(5).
050    005000     05  TABL-COND        PIC 9(5).
051    005100     05  TABL-ACTN        PIC 9(5).
052    005200     05  TABL-ITMS        PIC 9(5).
053    005300     05  TABL-ACTS        PIC 9(5).
054    005400     05  TABL-CNDS        PIC 9(5).
055    005500     05  TABL-RULS        PIC 9(5).
056    005600
```

```
005700 FD  DETABLS
005800     BLOCK CONTAINS 1 RECORDS
005900     RECORD CONTAINS 132 CHARACTERS
006000     DATA RECORD IS DTBL-NTRY
006100     LABEL RECORDS ARE OMITTED.
006200
006300 01  DTBL-NTRY.
006400     05  NTRY-COMN.
006500         10  NTRY-INDX    PIC 9(5).
006600         10  NTRY-PREV    PIC 9(5).
006700         10  NTRY-NEXT    PIC 9(5).
006800     05  NTRY-DATA        PIC X(117).
006900
007000 COMMON-STORAGE SECTION.
007100 01  COMN-INFO.
007200     05  RETN-CODE        PIC S9(4) COMP.
007300     05  PASD-INFO        PIC X(138).
007400
007500 WORKING-STORAGE SECTION.
007600 77  INDX-GP01            PIC 9(4)   COMP   VALUE 0.
007700 77  LAST-CODE            PIC X             VALUE SPACE.
007800 77  EROR-CODE            PIC S9(4)  COMP   VALUE +0.
007900 77  FRST-FREE            PIC 9(5)   COMP   VALUE 1.
008000 77  TOTL-TBLS            PIC 9(4)   COMP   VALUE 0.
008100 77  TRCE-CUNT            PIC S9(4)         VALUE +0.
008200 77  FILE-LNTH            PIC 9(5)          VALUE 2000.
008300 01  REC-KEY              PIC X(5)          VALUE LOW-VALUES.
008400 01  TBL-KEY              PIC 9(5)          VALUE 0.
008500 01  EROR-AREA.
008600     05  MESG-IDNT        PIC X(9)          VALUE ≠**** DET10≠.
008700     05  MESG-NMBR        PIC 99            VALUE 0.
008800     05  MESG-FL05        PIC X(5)          VALUE ≠ --- ≠.
008900     05  MESG-AREA        PIC X(117)        VALUE SPACE.
009000 01  NEXT-PNTR.
009100     05  NEXT-NMBR        PIC X(5)          VALUE ≠FIRST≠.
009200     05  NEXT-NEXT        PIC X(5)          VALUE LOW-VALUES.
009300     05  NEXT-LAST        PIC X(5)          VALUE LOW-VALUES.
009400     05  NEXT-INDX        PIC 9(5)          VALUE 0.
009500     05  NEXT-RFTO        PIC 9(5)          VALUE 0.
009600     05  NEXT-RFBY        PIC 9(5)          VALUE 0.
009700     05  NEXT-COND        PIC 9(5)          VALUE 0.
009800     05  NEXT-ACTN        PIC 9(5)          VALUE 0.
009900     05  NEXT-ITMS        PIC 9(5)          VALUE 0.
010000     05  NEXT-ACTS        PIC 9(5)          VALUE 0.
010100     05  NEXT-CNDS        PIC 9(5)          VALUE 0.
010200     05  NEXT-RULS        PIC 9(5)          VALUE 0.
010300
010400 01  EROR-MSGS.
010500     05  EROR-TABL.
010600       10  MSG01 PIC X(30) VALUE ≠NO OF TABLES GREATER THAN 1000≠.
010700       10  MSG02 PIC X(30) VALUE ≠INVALID RECORD CODE            ≠.
010800       10  MSG03 PIC X(30) VALUE ≠NO OF ENTRIES MORE THAN 2000   ≠.
010900       10  MSG04 PIC X(30) VALUE ≠FIRST POINTER MISSING          ≠.
011000       10  MSG05 PIC X(30) VALUE ≠REFERS-TO RECORD NOT FOUND     ≠.
011100       10  MSG06 PIC X(30) VALUE ≠TABLE DOES NOT EXIST           ≠.
011200       10  MSG07 PIC X(30) VALUE ≠REFERENCED-BY RECORD NOT FOUND≠.
```

```
113     011300          10  MSG08 PIC X(30) VALUE ≠POINTER LOST                    ≠.
114     011400          10  MSG09 PIC X(30) VALUE ≠UNABLE TO COMPLETE AVAIL CHAIN≠.
115     011500          10  MSG10 PIC X(30) VALUE ≠TABLE GENERATION BEGINS        ≠.
116     011600      05  EROR-MESG REDEFINES EROR-TABL OCCURS 10 TIMES PIC X(30).
117     011700
118     011800 01   NAME-RECD.
119     011900      05  NAME-COMN.
120     012000          10  NAME-INDX    PIC 9(5)        VALUE 0.
121     012100          10  NAME-PREV    PIC 9(5)        VALUE 0.
122     012200          10  NAME-NEXT    PIC 9(5)        VALUE 0.
123     012300      05  NAME-BODY        PIC X(117)      VALUE SPACE.
124     012400
125     012500 01   RFTO-RECD.
126     012600      05  RFTO-COMN.
127     012700          10  RFTO-INDX    PIC 9(5)        VALUE 0.
128     012800          10  RFTO-PREV    PIC 9(5)        VALUE 0.
129     012900          10  RFTO-NEXT    PIC 9(5)        VALUE 0.
130     013000      05  RFTO-NUMB        PIC 99          VALUE 0.
131     013100      05  RFTO-BODY        PIC X(115)      VALUE SPACE.
132     013200      05  RFTO-TBLE REDEFINES RFTO-BODY OCCURS 23 TIMES PIC X(5).
133     013300
134     013400 01   COND-RECD.
135     013500      05  COND-COMN.
136     013600          10  COND-INDX    PIC 9(5)        VALUE 0.
137     013700          10  COND-PREV    PIC 9(5)        VALUE 0.
138     013800          10  COND-NEXT    PIC 9(5)        VALUE 0.
139     013900      05  COND-BODY        PIC X(117)      VALUE SPACE.
140     014000      05  COND-FL01 REDEFINES COND-BODY.
141     014100          10  COND-INFO    PIC X(77).
142     014200          10  COND-RULE    PIC X       OCCURS 40 TIMES.
143     014300
144     014400 01   ACTN-RECD.
145     014500      05  ACTN-COMN.
146     014600          10  ACTN-INDX    PIC 9(5)        VALUE 0.
147     014700          10  ACTN-PREV    PIC 9(5)        VALUE 0.
148     014800          10  ACTN-NEXT    PIC 9(5)        VALUE 0.
149     014900      05  ACTN-BODY        PIC X(117)      VALUE SPACE.
150     015000      05  ACTN-FL01 REDEFINES ACTN-BODY.
151     015100          10  ACTN-INFO    PIC X(77).
152     015200          10  ACTN-RULE    PIC X       OCCURS 40 TIMES.
153     015300
154     015400 01   PNTR-RECD.
155     015500      05  PNTR-IDNT        PIC X(5)        VALUE SPACE.
156     015600      05  PNTR-NEXT        PIC X(5)        VALUE SPACE.
157     015700      05  PNTR-FL05        PIC X(10)       VALUE SPACE.
158     015800      05  PNTR-RFTO        PIC 9(5)        VALUE 0.
159     015900      05  PNTR-FL10        PIC X(35)       VALUE SPACE.
160     016000
161     016100 01   RFBY-RECD.
162     016200      05  RFBY-COMN.
163     016300          10  RFBY-INDX    PIC 9(5)        VALUE 0.
164     016400          10  RFBY-PREV    PIC 9(5)        VALUE 0.
165     016500          10  RFBY-NEXT    PIC 9(5)        VALUE 0.
166     016600      05  RFBY-NUMB        PIC 99          VALUE 0.
167     016700      05  RFBY-BODY        PIC X(115)      VALUE SPACE.
168     016800      05  RFBY-TBLE REDEFINES RFBY-BODY OCCURS 23 TIMES PIC X(5).
```

```
169      016900
170      017000 01  TABL-REFD.
171      017100       05  REFD-FL05          PIC X(25)        VALUE SPACE.
172      017200       05  REFD-RFRY          PIC 9(5)         VALUE 0.
173      017300       05  REFD-FL10          PIC X(30)        VALUE SPACE.
174      017400
175      017500 01  TRCE-AREA               PIC X(132)       VALUE SPACE.
```

```
017600 PROCEDURE DIVISION USING COMN-INFO.
017700 000-PROGRAM-START SECTION 65.
017800 100-START.
017900     MOVE +10 TO EROR-CODE.
018000     PERFORM 600-WRITE-MESG THRU 699-WRITE-MESG-END.
018100     OPEN INPUT FRMTDFLE.
018200     OPEN I-O   DTINDEX.
018300     OPEN I-O   DETABLS.
018400     MOVE NEXT-PNTR TO DTBL-PNTR.
018500     GO TO 210-READ.
018600
018700 200-READ.
018800     MOVE RECD-CODE TO LAST-CODE.
018900
019000 210-READ.
019100     READ FRMTDFLE AT END
019200         GO TO 400-CHECK-REFERENCES.
019300
019400 220-CHECK-TYPE.
019500     IF RECD-CODE = ≠A≠ OR ≠R≠ OR ≠E≠
019600         GO TO 300-ACTION.
019700     IF RECD-CODE = ≠C≠
019800         GO TO 320-CONDITION.
019900     IF RECD-CODE = ≠N≠
020000         GO TO 340-NEW-TABLE.
020100     IF RECD-CODE = ≠,≠ OR ≠-≠
020200         GO TO 360-SAME-AS-LAST.
020300     MOVE +2 TO EROR-CODE.
020400     GO TO 950-ERROR-TERM.
020500
020600 300-ACTION.
020700     PERFORM 380-CHECK-GO THRU 399-CHECK-GO-END.
020800     ADD 1 TO TABL-ACTS.
020900     ADD 1 TO TABL-ITMS.
021000     IF TABL-RULS = 0
021100         MOVE 1 TO TABL-RULS.
021200     IF ACTN-INDX = 0
021300         MOVE FRST-FREE TO TABL-ACTN
021400         GO TO 305-FIRST-TIME.
021500     MOVE ACTN-RECD TO DTBL-NTRY.
021600     MOVE NTRY-INDX TO ACTN-PREV.
021700     MOVE FRST-FREE TO NTRY-NEXT.
021800     PERFORM 700-WRITE-TABLE THRU 799-WRITE-TABLE-END.
021900
022000 305-FIRST-TIME.
022100     MOVE FRST-FREE TO ACTN-INDX.
022200     ADD +1 TO FRST-FREE.
022300     MOVE RECD-BODY TO ACTN-BODY.
022400     IF RECD-CODE = ≠E≠
022500         MOVE ≠X≠ TO ACTN-RULE (2)
022600     ELSE
022700         MOVE ≠X≠ TO ACTN-RULE (1).
022800     GO TO 200-READ.
022900
023000 320-CONDITION.
023100     ADD 1 TO TABL-CNDS.
```

```
232     023200      ADD 1 TO TABL-ITMS.
233     023300      IF TABL-RULS IS LESS THAN 2
234     023400          MOVE 2 TO TABL-RULS.
235     023500      IF COND-INDX = 0
236     023600          MOVE FRST-FREE TO TABL-COND
237     023700          GO TO 325-FIRST-TIME.
238     023800      MOVE COND-RECD TO DTBL-NTRY.
239     023900      MOVE NTRY-INDX TO COND-PREV.
240     024000      MOVE FRST-FREE TO NTRY-NEXT.
241     024100      PERFORM 700-WRITE-TABLE THRU 799-WRITE-TABLE-END.
242     024200
243     024300 325-FIRST-TIME.
244     024400      MOVE FRST-FREE TO COND-INDX.
245     024500      ADD +1 TO FRST-FREE.
246     024600      MOVE RECD-BODY TO COND-INFO.
247     024700      MOVE ≠Y≠ TO COND-RULE (1).
248     024800      MOVE ≠N≠ TO COND-RULE (2).
249     024900      GO TO 200-READ.
250     025000
251     025100 340-NEW-TABLE.
252     025200      IF NAME-INDX = 0
253     025300          GO TO 342-FIRST-TIME.
254     025400      MOVE NAME-RECD TO DTBL-NTRY.
255     025500      MOVE 0 TO NAME-PREV.
256     025600      PERFORM 700-WRITE-TABLE THRU 799-WRITE-TABLE-END.
257     025700
258     025800 342-FIRST-TIME.
259     025900      MOVE FRST-FREE TO NAME-INDX.
260     026000      ADD +1 TO FRST-FREE.
261     026100      MOVE RECD-BODY TO NAME-BODY.
262     026200      ADD +1 TO TOTL-TBLS.
263     026300      IF RFTO-INDX = 0
264     026400          GO TO 344-CHECK-COND.
265     026500      MOVE RFTO-RECD TO DTBL-NTRY.
266     026600      PERFORM 700-WRITE-TABLE THRU 799-WRITE-TABLE-END.
267     026700      MOVE 0 TO RFTO-INDX RFTO-PREV.
268     026800      MOVE 0 TO RFTO-NUMB.
269     026900      MOVE SPACE TO RFTO-BODY.
270     027000
271     027100 344-CHECK-COND.
272     027200      IF COND-INDX = 0
273     027300          GO TO 346-CHECK-ACTION.
274     027400      MOVE COND-RECD TO DTBL-NTRY.
275     027500      PERFORM 700-WRITE-TABLE THRU 799-WRITE-TABLE-END.
276     027600      MOVE 0 TO COND-INDX COND-PREV.
277     027700
278     027800 346-CHECK-ACTION.
279     027900      IF ACTN-INDX = 0
280     028000          GO TO 348-WRITE-INDX.
281     028100      MOVE ACTN-RECD TO DTBL-NTRY.
282     028200      PERFORM 700-WRITE-TABLE THRU 799-WRITE-TABLE-END.
283     028300      MOVE 0 TO ACTN-INDX ACTN-PREV.
284     028400
285     028500 348-WRITE-INDX.
286     028600      MOVE RECD-BODY TO TABL-NEXT NEXT-NMBR.
287     028700      PERFORM 800-WRITE-INDEX THRU 899-WRITE-INDEX-END.
```

```
0288     028800      MOVE NAME-INDX TO TABL-INDX.
0289     028900      GO TO 200-READ.
0290     029000
0291     029100 350-CONTINUED-NAME.
0292     029200      ADD 1 TO TABL-ITMS.
0293     029300      MOVE NAME-RECD TO DTBL-NTRY.
0294     029400      MOVE NTRY-INDX TO NAME-PREV.
0295     029500      MOVE FRST-FREE TO NTRY-NEXT.
0296     029600      PERFORM 700-WRITE-TABLE THRU 799-WRITE-TABLE-END.
0297     029700      MOVE FRST-FREE TO NAME-INDX.
0298     029800      ADD +1 TO FRST-FREE.
0299     029900      MOVE RECD-BODY TO NAME-BODY.
0300     030000      GO TO 200-READ.
0301     030100
0302     030200 360-SAME-AS-LAST.
0303     030300      IF LAST-CODE = ≠R≠ OR ≠E≠
0304     030400          MOVE ≠A≠ TO RECD-CODE
0305     030500      ELSE
0306     030600          MOVE LAST-CODE TO RECD-CODE.
0307     030700      IF RECD-CODE = ≠N≠
0308     030800          GO TO 350-CONTINUED-NAME.
0309     030900      GO TO 220-CHECK-TYPE.
0310     031000
0311     031100 380-CHECK-GO.
0312     031200      IF RECD-GOTO = ≠GO TO ≠
0313     031300          GO TO 385-GO-STATEMENT.
0314     031400      GO TO 399-CHECK-GO-END.
0315     031500
0316     031600 385-GO-STATEMENT.
0317     031700      IF RFTO-INDX = 0
0318     031800          MOVE FRST-FREE TO TABL-RFTO
0319     031900          GO TO 390-FIRST-TIME.
0320     032000      IF RFTO-NUMB IS LESS THAN 23
0321     032100          GO TO 392-ENTER-DATA.
0322     032200      MOVE RFTO-RECD TO DTBL-NTRY.
0323     032300      MOVE SPACE     TO RFTO-BODY.
0324     032400      MOVE NTRY-INDX TO RFTO-PREV.
0325     032500      MOVE FRST-FREE TO NTRY-NEXT.
0326     032600      PERFORM 700-WRITE-TABLE THRU 799-WRITE-TABLE-END.
0327     032700      MOVE 0 TO RFTO-NUMB.
0328     032800
0329     032900 390-FIRST-TIME.
0330     033000      MOVE FRST-FREE TO RFTO-INDX.
0331     033100      ADD +1 TO FRST-FREE.
0332     033200
0333     033300 392-ENTER-DATA.
0334     033400      MOVE 0 TO INDX-GP01.
0335     033500      PERFORM 394-CHECK-FOR-DUP THRU 396-CHECK-FOR-DUP-END
0336     033600          RFTO-NUMB TIMES.
0337     033700      ADD +1 TO RFTO-NUMB.
0338     033800      MOVE RECD-NAME TO RFTO-TBLE (RFTO-NUMB).
0339     033900      GO TO 399-CHECK-GO-END.
0340     034000
0341     034100 394-CHECK-FOR-DUP.
0342     034200      ADD +1 TO INDX-GP01.
0343     034300      IF RECD-NAME = RFTO-TBLE (INDX-GP01)
```

```
034400          GO TO 399-CHECK-GO-END.
034500
034600 396-CHECK-FOR-DUP-END.
034700     EXIT.
034800
034900 399-CHECK-GO-END.
035000     EXIT.
035100
035200 400-CHECK-REFERENCES.
035300     MOVE ≠AVAIL≠ TO TABL-NEXT NEXT-NMBR.
035400     PERFORM 800-WRITE-INDEX THRU 899-WRITE-INDEX-END.
035500     MOVE ≠99999≠ TO TABL-NEXT.
035600     PERFORM 500-CLEAN-UP THRU 599-CLEAN-UP-END.
035700     MOVE TOTL-TBLS TO TABL-INDX.
035800     PERFORM 800-WRITE-INDEX THRU 899-WRITE-INDEX-END.
035900     MOVE ≠FIRST≠ TO REC-KEY.
036000     READ DTINDEX INTO PNTR-RECD INVALID KEY
036100         MOVE +4 TO EROR-CODE
036200         GO TO 950-ERROR-TERM.
036300     MOVE PNTR-NEXT TO REC-KEY.
036400     PERFORM 433-FIRST-TABLE THRU 499-REFERENCES-END.
036500
036600 410-READ-POINTER.
036700     MOVE PNTR-NEXT TO REC-KEY.
036800     READ DTINDEX INTO PNTR-RECD INVALID KEY
036900         GO TO 900-TERMINATION.
037000     MOVE PNTR-RFTO TO TBL-KEY.
037100
037200 420-READ-REFRECD.
037300     READ DETABLS INTO RFTO-RECD INVALID KEY
037400         MOVE +5 TO EROR-CODE
037500         GO TO 410-READ-POINTER.
037600     MOVE +0 TO INDX-GP01.
037700     PERFORM 430-REFERENCES THRU 499-REFERENCES-END RFTO-NUMR
037800         TIMES.
037900     IF RFTO-NEXT = 0
038000         GO TO 410-READ-POINTER.
038100     GO TO 420-READ-REFRECD.
038200
038300 430-REFERENCES.
038400     ADD +1 TO INDX-GP01.
038500     MOVE RFTO-TBLE (INDX-GP01) TO REC-KEY.
038600
038700 433-FIRST-TABLE.
038800     READ DTINDEX INTO TABL-REFD INVALID KEY
038900         MOVE +6 TO EROR-CODE
039000         GO TO 950-ERROR-TERM.
039100     IF REFD-RFBY = 0
039200         MOVE FRST-FREE TO REFD-RFBY
039300         MOVE TABL-REFD TO DTBL-PNTR
039400         PERFORM 850-WRITE-RECORD THRU 860-WRITE-RECORD-END
039500         GO TO 440-NEW-RECORD.
039600     MOVE REFD-RFBY TO TBL-KEY.
039700
039800 435-READ-RFBY.
039900     READ DETABLS INTO RFBY-RECD INVALID KEY
```

```
0400    040000          MOVE +7 TO EROR-CODE
0401    040100          GO TO 950-ERROR-TERM.
0402    040200      IF RFBY-NUMB IS LESS THAN 23
0403    040300          GO TO 450-GEN-RFBY.
0404    040400      IF RFBY-NEXT NOT = 0
0405    040500          MOVE RFBY-NEXT TO TBL-KEY
0406    040600          GO TO 435-READ-RFBY.
0407    040700      MOVE FRST-FREE TO RFBY-NEXT.
0408    040800      MOVE RFBY-RECD TO DTBL-NTRY.
0409    040900      PERFORM 700-WRITE-TABLE THRU 799-WRITE-TABLE-END.
0410    041000      MOVE RFBY-INDX TO RFBY-PREV.
0411    041100      GO TO 445-INITIALIZE.
0412    041200
0413    041300 440-NEW-RECORD.
0414    041400      MOVE 0          TO RFBY-PREV.
0415    041500
0416    041600 445-INITIALIZE.
0417    041700      MOVE 0          TO RFBY-NEXT.
0418    041800      MOVE 1          TO RFBY-NUMB.
0419    041900      MOVE SPACE      TO RFBY-BODY.
0420    042000      MOVE PNTR-IDNT TO RFBY-TBLE (1).
0421    042100      MOVE FRST-FREE TO RFBY-INDX TBL-KEY.
0422    042200      ADD +1          TO FRST-FREE.
0423    042300      GO TO 460-WRITE.
0424    042400
0425    042500 450-GEN-RFBY.
0426    042600      ADD 1 TO RFBY-NUMB.
0427    042700      MOVE PNTR-IDNT TO RFBY-TBLE (RFBY-NUMB).
0428    042800
0429    042900 460-WRITE.
0430    043000      MOVE RFBY-RECD TO DTBL-NTRY.
0431    043100      PERFORM 700-WRITE-TABLE THRU 799-WRITE-TABLE-END.
0432    043200      GO TO 499-REFERENCES-END.
0433    043300
0434    043400 499-REFERENCES-END.
0435    043500      EXIT.
0436    043600
0437    043700 500-CLEAN-UP.
0438    043800      IF NAME-INDX = 0
0439    043900          GO TO 510-RFTO.
0440    044000      MOVE NAME-RECD TO DTBL-NTRY.
0441    044100      PERFORM 700-WRITE-TABLE THRU 799-WRITE-TABLE-END.
0442    044200
0443    044300 510-RFTO.
0444    044400      IF RFTO-INDX = 0
0445    044500          GO TO 520-COND.
0446    044600      MOVE RFTO-RECD TO DTBL-NTRY.
0447    044700      PERFORM 700-WRITE-TABLE THRU 799-WRITE-TABLE-END.
0448    044800
0449    044900 520-COND.
0450    045000      IF COND-INDX = 0
0451    045100          GO TO 530-ACTN.
0452    045200      MOVE COND-RECD TO DTBL-NTRY.
0453    045300      PERFORM 700-WRITE-TABLE THRU 799-WRITE-TABLE-END.
0454    045400
0455    045500 530-ACTN.
```

```
456      045600      IF ACTN-INDX = 0
457      045700          GO TO 599-CLEAN-UP-END.
458      045800      MOVE ACTN-RECD TO DTBL-NTRY.
459      045900      PERFORM 700-WRITE-TABLE THRU 799-WRITE-TABLE-END.
460      046000
461      046100  599-CLEAN-UP-END.
462      046200      EXIT.
463      046300
464      046400  600-WRITE-MESG.
465      046500      MOVE EROR-MESG (EROR-CODE) TO MESG-AREA.
466      046600      MOVE EROR-CODE TO MESG-NMBR.
467      046700      DISPLAY EROR-AREA.
468      046800
469      046900  699-WRITE-MESG-END.
470      047000      EXIT.
471      047100
472      047200  700-WRITE-TABLE.
473      047300      MOVE NTRY-INDX TO TBL-KEY.
474      047400      WRITE DTBL-NTRY INVALID KEY
475      047500          MOVE +3 TO EROR-CODE
476      047600          GO TO 950-ERROR-TERM.
477      047700
478      047800  799-WRITE-TABLE-END.
479      047900      EXIT.
480      048000
481      048100  800-WRITE-INDEX.
482      048200      MOVE TABL-NMBR TO REC-KEY NEXT-LAST.
483      048300
484      048400  850-WRITE-RECORD.
485      048500      WRITE DTBL-PNTR INVALID KEY
486      048600          MOVE +1 TO EROR-CODE
487      048700          GO TO 950-ERROR-TERM.
488      048800
489      048900  860-WRITE-RECORD-END.
490      049000      EXIT.
491      049100
492      049200  870-NEXT-PNTR.
493      049300      MOVE NEXT-PNTR TO DTBL-PNTR.
494      049400
495      049500  899-WRITE-INDEX-END.
496      049600      EXIT.
497      049700
498      049800  900-TERMINATION.
499      049900      MOVE ≠AVAIL≠ TO REC-KEY.
500      050000      READ DTINDEX INVALID KEY
501      050100          MOVE +8 TO EROR-CODE
502      050200          GO TO 950-ERROR-TERM.
503      050300      MOVE FRST-FREE TO TABL-RFTO.
504      050400      MOVE FILE-LNTH TO TABL-RFBY.
505      050500      WRITE DTBL-PNTR INVALID KEY
506      050600          MOVE +8 TO EROR-CODE
507      050700          GO TO 950-ERROR-TERM.
508      050800      MOVE FRST-FREE TO NAME-INDX TBL-KEY.
509      050900      MOVE 0         TO NAME-PREV.
510      051000      MOVE SPACE     TO NAME-BODY.
511      051100
```

```
512      051200 920-WRITE-FREE.
513      051300     IF FRST-FREE IS GREATER THAN FILE-LNTH
514      051400         GO TO 998-ENDIT
515      051500     ELSE
516      051600         ADD 1 TO FRST-FREE
517      051700         MOVE FRST-FREE TO NAME-NEXT.
518      051800     MOVE NAME-RECD TO DTBL-NTRY.
519      051900     WRITE DTBL-NTRY FROM NAME-RECD INVALID KEY
520      052000         MOVE +9 TO EROR-CODE
521      052100         GO TO 950-ERROR-TERM.
522      052200     MOVE NAME-INDX TO NAME-PREV.
523      052300     MOVE NAME-NEXT TO NAME-INDX TBL-KEY.
524      052400     GO TO 920-WRITE-FREE.
525      052500
526      052600 950-ERROR-TERM.
527      052700     PERFORM 600-WRITE-MESG THRU 699-WRITE-MESG-END.
528      052800     MOVE +99 TO RETN-CODE.
529      052900     GO TO 999-ENDIT.
530      053000
531      053100 998-ENDIT.
532      053200     MOVE +0 TO RETN-CODE.
533      053300
534      053400 999-ENDIT.
535      053500     CLOSE FRMTOFLE.
536      053600     CLOSE DTINDEX.
537      053700     CLOSE DETABLS.
538      053800     RETURN.
539      053900
540      054000
0 LENGTH IS  002333
0R SCM USED
```

```
0001    000100 IDENTIFICATION DIVISION.
0002    000200 PROGRAM-ID. DETRL20.
0003    000300 INSTALLATION. SIR GEORGE WILLIAMS UNIVERSITY.
0004    000400 DATE-COMPILED. MAY 21,  75/07/18..
0005    000500 REMARKS.
0006    000600          THIS PROGRAM WILL COMBINE ONE TABLE WITH ANOTHER UNDER
0007    000700      THE FOLLOWING CONDITIONS
0008    000800
0009    000900      ------------------------------------------------------
0010    001000      I 1) THE MERGED TABLE IS NOT REFERENCED BY    I Y I Y I E I
0011    001100      I    OTHER TABLES                             I   I   I   I
0012    001200      I---------------------------------------------I---I---I---I
0013    001300      I 2) THE MERGED TABLE IS AN ACTION TABLE      I Y I N I L I
0014    001400      I---------------------------------------------I---I---I---I
0015    001500      I 3) THE RECEIVING TABLE IS AN ACTION TABLE   I - I N I S I
0016    001600      I---------------------------------------------I---I---I---I
0017    001700      I 4) THE RECEIVING TABLE CONTAINS ACTIONS IN  I - I N I E I
0018    001800      I    THE REFERRING RULE BESIDES THE GO TO     I   I   I   I
0019    001900      I=============================================I===I===I===I
0020    002000      I    MERGE                                    o X I X I   I
0021    002100      I    DONT MERGE                               I   I   I X I
0022    002200      I---------------------------------------------------------I.
0023    002300 ENVIRONMENT DIVISION.
0024    002400 CONFIGURATION SECTION.
0025    002500 SOURCE-COMPUTER. 6400.
0026    002600 OBJECT-COMPUTER. 6400.
0027    002700 INPUT-OUTPUT SECTION.
0028    002800 FILE-CONTROL.
0029    002900     SELECT DTINDEX ASSIGN TO DTINDEX
0030    003000         ORGANIZATION IS STANDARD
0031    003100         FILE-LIMIT IS 1000
0032    003200         ACCESS MODE IS RANDOM
0033    003300         SYMBOLIC KEY IS INDX-KEY.
0034    003400
0035    003500     SELECT DETABLS ASSIGN TO DECTBLS
0036    003600         FILE-LIMIT IS 2000
0037    003700         ACCESS MODE IS RANDOM
0038    003800         ACTUAL KEY IS TABL-KEY.
0039    003900 DATA DIVISION.
0040    004000 FILE SECTION.
0041    004100
0042    004200 FD  DTINDEX
0043    004300     BLOCK CONTAINS 1 RECORDS
0044    004400     RECORD CONTAINS 60 CHARACTERS
0045    004500     DATA RECORD IS DTBL-PNTR
0046    004600     LABEL RECORDS ARE OMITTED.
0047    004700
0048    004800 01  DTBL-PNTR.
0049    004900     05  TABL-NMBR       PIC X(5).
0050    005000     05  TABL-NEXT       PIC X(5).
0051    005100     05  TABL-LAST       PIC X(5).
0052    005200     05  TABL-INDX       PIC 9(5).
0053    005300     05  TABL-RFTO       PIC 9(5).
0054    005400     05  TABL-RFBY       PIC 9(5).
0055    005500     05  TABL-COND       PIC 9(5).
0056    005600     05  TABL-ACTN       PIC 9(5).
```

```
005700      05  TABL-ITMS          PIC 9(5).
005800      05  TABL-ACTS          PIC 9(5).
005900      05  TABL-CNDS          PIC 9(5).
006000      05  TABL-RULS          PIC 9(5).
006100
006200 FD  DETABLS
006300      BLOCK CONTAINS 1 RECORDS
006400      RECORD CONTAINS 132 CHARACTERS
006500      DATA RECORD IS DTBL-NTRY
006600      LABEL RECORDS ARE OMITTED.
006700
006800 01  DTBL-NTRY.
006900      05  NTRY-COMN.
007000          10  NTRY-INDX      PIC 9(5).
007100          10  NTRY-PREV      PIC 9(5).
007200          10  NTRY-NEXT      PIC 9(5).
007300      05  NTRY-DATA          PIC X(117).
007400
007500 COMMON-STORAGE SECTION.
007600 01  COMN-INFO.
007700      05  RETN-CODE          PIC S9(4)        COMP.
007800      05  PASD-INFO.
007900          10  MRGE-TRIS      PIC 9.
008000          10  PASD-FLDS      PIC X(137).
008100
008200 WORKING-STORAGE SECTION.
008300 77  INDX-KEY                PIC X(5)         VALUE ≠FIRST≠.
008400 77  TABL-KEY                PIC 9(5)         VALUE 0.
008500 77  ERDR-CODE               PIC 9(4) COMP    VALUE 0.
008600 77  NOOF-PSES               PIC 9(4)         VALUE 1.
008700 77  INDX-GP01               PIC 9(4) COMP    VALUE 0.
008800 77  INDX-GP02               PIC 9(4) COMP    VALUE 0.
008900 77  MAXM-PSES               PIC 9(4) COMP    VALUE 1.
009000 77  MRGE-TEST               PIC XXX          VALUE ≠NO ≠.
009100 77  NEXT-INDX               PIC 9(5)         VALUE 0.
009200 77  LAST-INDX               PIC 9(5)         VALUE 0.
009300 77  CHCK-REQD               PIC 9            VALUE 0.
009400 77  INDX-GP03               PIC 9(4) COMP    VALUE 0.
009500 77  INDX-GP04               PIC 9(4) COMP    VALUE 0.
009600 77  INDX-GP05               PIC 9(4) COMP    VALUE 0.
009700 77  INDX-GP06               PIC 9(4) COMP    VALUE 0.
009800 77  INDX-GP07               PIC 9(4) COMP    VALUE 0.
009900 77  INDX-GP08               PIC 9(4) COMP    VALUE 0.
010000 77  INDX-GP09               PIC 9(4) COMP    VALUE 0.
010100 77  INDX-GP10               PIC 9(4) COMP    VALUE 0.
010200 77  INDX-GP11               PIC 9(4) COMP    VALUE 0.
010300 77  TOTL-RULS               PIC 9(4) COMP    VALUE 0.
010400 77  LAST-PCRD               PIC 9(5)         VALUE 0.
010500 77  TEST-TABL               PIC X(5)         VALUE SPACE.
010600 77  MRGE-FLAG               PIC X            VALUE SPACE.
010700 77  RLSE-SNGL               PIC X            VALUE SPACE.
010800 77  SAVE-NEXT               PIC X(5)         VALUE SPACE.
010900 77  SAVE-PREV               PIC X(5)         VALUE SPACE.
011000 77  TBLS-MRGD               PIC X            VALUE SPACE.
011100
011200 01  ERDR-AREA.
```

```
011300      05  MESG-IDNT          PIC X(9)        VALUE ≠*** DET20≠.
011400      05  MESG-NMBR          PIC 99          VALUE 0.
011500      05  MESG-FL05          PIC X(5)        VALUE ≠ --- ≠.
011600      05  EROR-TEXT          PIC X(30)       VALUE SPACE.
011700      05  EROR-FL05          PIC X           VALUE SPACE.
011800      05  EROR-VALU          PIC X(84)       VALUE SPACE.
011900
012000  01  EROR-MSGS.
012100      05  EROR-TABL.
012200        10  MSG01 PIC X(30) VALUE ≠TABLE NOT FOUND - TERMINATE   ≠.
012300        10  MSG02 PIC X(30) VALUE ≠NO ENTRY IN FILE - SKIP       ≠.
012400        10  MSG03 PIC X(30) VALUE ≠UNABLE TO WRITE ENTRY - SKIP  ≠.
012500        10  MSG04 PIC X(30) VALUE ≠UNABLE TO WRITE TABLE - TERMIN≠.
012600        10  MSG05 PIC X(30) VALUE ≠PROGRAM ERROR ENCOUNTERED LOC ≠.
012700        10  MSG06 PIC X(30) VALUE ≠TABLE MERGE REQUESTED         ≠.
012800        10  MSG07 PIC X(30) VALUE ≠FURTHER MERGE USELESS PASSES =≠.
012900        10  MSG08 PIC X(30) VALUE ≠COMBINING TABLES ---          ≠.
013000        10  MSG09 PIC X(30) VALUE ≠MERGE PHASE PASS NUMBER ---   ≠.
013100      05  EROR-MESG REDEFINES EROR-TABL OCCURS 9 TIMES PIC X(30).
013200
013300  01  TABL-LIST.
013400      05  NEXT-TABL          PIC X(5)        VALUE SPACE.
013500      05  MRGD-TABL          PIC 999V99      VALUE 0.
013600      05  MRGD-INDX REDEFINES MRGD-TABL PIC X(5).
013700      05  RCVG-TABL          PIC 999V99      VALUE 0.
013800      05  RCVG-INDX REDEFINES RCVG-TABL PIC X(5).
013900
014000  01  RULE-FLGS.
014100      05  RULE-FLAG          PIC X           OCCURS 40 TIMES.
014200
014300  01  GOTO-TABL.
014400      05  NUMB-GOTO          PIC 9(4) COMP   VALUE 0.
014500      05  GOTO-RECD          PIC 9(5)        OCCURS 10 TIMES.
014600
014700  01  RCVG-PNTR.
014800      05  RCVG-NMBR          PIC X(5)        VALUE SPACE.
014900      05  RCVG-NUMB REDEFINES RCVG-NMBR PIC 999V99.
015000      05  RCVG-NEXT          PIC X(5)        VALUE SPACE.
015100      05  RCVG-LAST          PIC X(5)        VALUE SPACE.
015200      05  RCVG-INEX          PIC 9(5)        VALUE 0.
015300      05  RCVG-RFTO          PIC 9(5)        VALUE 0.
015400      05  RCVG-RFBY          PIC 9(5)        VALUE 0.
015500      05  RCVG-COND          PIC 9(5)        VALUE 0.
015600      05  RCVG-ACTN          PIC 9(5)        VALUE 0.
015700      05  RCVG-ITMS          PIC 9(5)        VALUE 0.
015800      05  RCVG-ACTS          PIC 9(5)        VALUE 0.
015900      05  RCVG-CNDS          PIC 9(5)        VALUE 0.
016000      05  RCVG-RULS          PIC 9(5)        VALUE 0.
016100  01  NAME-RECD.
016200      05  NAME-NMBR          PIC 999V99      VALUE 0.
016300      05  NAME-ORIG          PIC X(106)      VALUE SPACE.
016400
016500  01  COND-RECD.
016600      05  COND-INFO          PIC X(77)       VALUE SPACE.
016700      05  COND-FL01.
016800          10  COND-RULE      PIC X           OCCURS 40 TIMES.
```

```
169      016900
170      017000 01  ACTN-RECD.
171      017100      05  ACTN-INFO        PIC X(77)        VALUE SPACE.
172      017200      05  ACTN-GOTO REDEFINES ACTN-INFO.
173      017300          10  GOTO-CNST    PIC X(6).
174      017400          10  GOTO-NAME    PIC 999V99.
175      017500          10  GOTO-FL05    PIC X(66).
176      017600      05  ACTN-FL05.
177      017700          10  ACTN-RULE    PIC X            OCCURS 40 TIMES.
178      017800
179      017900 01  TEMP-AREA.
180      018000      05  TEMP-FL05        PIC X(92)        VALUE SPACE.
181      018100      05  TEMP-RULS.
182      018200          10  TEMP-RULE    PIC X            OCCURS 40 TIMES.
183      018300      05  TEMP-WK01        REDEFINES TEMP-RULS.
184      018400          10  TEMP-FL10    PIC X.
185      018500          10  TEMP-WK02    PIC X(39).
186      018600 01  RFTO-RECD.
187      018700      05  RFTO-NUMB        PIC 99           VALUE 0.
188      018800      05  RFTO-INFO.
189      018900          10  RFTO-TABL    PIC X(5)         OCCURS 23 TIMES.
190      019000
191      019100 01  RFBY-RECD.
192      019200      05  RFBY-NUMB        PIC 99           VALUE 0.
193      019300      05  RFBY-INFO.
194      019400          10  RFBY-TABL    PIC X(5)         OCCURS 23 TIMES.
195      019500
196      019600 01  RFTO-LIST.
197      019700      05  RFTO-LNTH        PIC 99           VALUE 0.
198      019800      05  RFTO-ITEM        PIC X(5)         OCCURS 100 TIMES.
199      019900
200      020000 01  RFBY-LIST.
201      020100      05  RFBY-LNTH        PIC 99           VALUE 0.
202      020200      05  RFBY-ITEM        PIC X(5) OCCURS 100 TIMES.
203      020300
204      020400 01  AVAL-PNTR.
205      020500      05  AVAL-NMBR        PIC X(5)         VALUE SPACE.
206      020600      05  AVAL-NEXT        PIC X(5)         VALUE SPACE.
207      020700      05  AVAL-LAST        PIC X(5)         VALUE SPACE.
208      020800      05  AVAL-TBLS        PIC 9(5)         VALUE 0.
209      020900      05  AVAL-RECD        PIC 9(5)         VALUE 0.
210      021000      05  AVAL-TERM        PIC 9(5)         VALUE 0.
211      021100      05  AVAL-FL05        PIC X(30)        VALUE SPACE.
212      021200
213      021300 01  DCSN-TABL.
214      021400      05  CNDS-RCVG        PIC S9(4) COMP   VALUE +0.
215      021500      05  CNDS-MRGD        PIC S9(4) COMP   VALUE +0.
216      021600      05  ACTS-RCVG        PIC S9(4) COMP   VALUE +0.
217      021700      05  ACTS-MRGD        PIC S9(4) COMP   VALUE +0.
218      021800      05  DCSN-ITMS OCCURS 100 TIMES.
219      021900          10  ITMS-CRNT    PIC 9(5).
220      022000          10  ITMS-PREV    PIC 9(5).
221      022100          10  ITMS-NEXT    PIC 9(5).
222      022200          10  ITMS-FL05    PIC X(117).
223      022300
224      022400 01  CORE-SPND.
```

```
225      022500      05  POSN-TABL                          OCCURS 40 TIMES.
226      022600          10  POSN-NIEW   PIC 99.
227      022700
228      022800  01  STUB-TABL.
229      022900      05  NMBR-STBS      PIC 99             COMP    VALUE 0.
230      023000      05  MRGD-STBS                         OCCURS 100 TIMES.
231      023100          10  MRGD-STUB  PIC X(40).
232      023200
233      023300  01  SHFT-AREA.
234      023400      05  AREA-FL05.
235      023500          10  SHFT-RULE  PIC X              OCCURS 40 TIMES.
236      023600
237      023700      05  SHFT-WHOL REDEFINES AREA-FL05.
238      023800          10  SHFT-FRST  PIC X.
239      023900          10  SHFT-REST  PIC X(39).
240      024000
241      024100  01  SAVE-PNTR.
242      024200      05  SAVE-KEY       PIC X(5).
243      024300      05  SAVE-REST      PIC X(55).
244      024400
245      024500  01  GOTO-LINE.
246      024600      05  GOTO-FXED      PIC X(77)          VALUE SPACE.
247      024700      05  GOTO-RULE      PIC X OCCURS 40 TIMES.
248      024800
249      024900  01  GOTO-AREA.
250      025000      05  GOTO-PREV      PIC 9(5)           VALUE 0.
251      025100      05  GOTO-NEXT      PIC 9(5)           VALUE 0.
252      025200      05  GOTO-THIS      PIC 9(5)           VALUE 0.
253      025300
254      025400  01  TABN-AREA.
255      025500      05  TAB1-NAME      PIC ZZ9.99.
256      025600      05  TABN-FL05      PIC X(5)           VALUE ≠ AND ≠.
257      025700      05  TAB2-NAME      PIC ZZ9.99.
258      025800
```

```
259      025900 PROCEDURE DIVISION USING COMN-INFO.
260      026000 000-PROGRAM-START SECTION 70.
261      026100 000-START.
262      026200     MOVE MRGE-TRIS TO MAXM-PSES.
263      026300     MOVE +6 TO EROR-CODE
264      026400     PERFORM 600-ERROR-ROUTINE THRU 699-ERROR-ROUTINE-END.
265      026500     OPEN I-O DTINDEX.
266      026600     OPEN I-O DETABLS.
267      026700     MOVE ≠AVAIL≠ TO INDX-KEY.
268      026800     PERFORM 700-READ-POINTER THRU 749-READ-POINTER-END.
269      026900     MOVE DTBL-PNTR TO AVAL-PNTR.
270      027000     MOVE ≠FIRST≠ TO INDX-KEY.
271      027100     MOVE NOOF-PSES TO EROR-VALU.
272      027200     MOVE 9 TO EROR-CODE.
273      027300     PERFORM 600-ERROR-ROUTINE THRU 699-ERROR-ROUTINE-END.
274      027400
275      027500 100-READ-TABLE.
276      027600     PERFORM 700-READ-POINTER THRU 749-READ-POINTER-END.
277      027700
278      027800 110-READ-TABLE.
279      027900     IF TABL-NEXT NOT = ≠AVAIL≠
280      028000         GO TO 120-READ-TABLE.
281      028100     ADD 1 TO NOOF-PSES.
282      028200     IF NOOF-PSES IS GREATER THAN MAXM-PSES
283      028300         GO TO 900-TERMINATION.
284      028400     MOVE 9 TO EROR-CODE.
285      028500     MOVE NOOF-PSES TO EROR-VALU.
286      028600     PERFORM 600-ERROR-ROUTINE THRU 699-ERROR-ROUTINE-END.
287      028700         GO TO 900-TERMINATION.
288      028800     IF TBLS-MRGD = SPACE
289      028900         GO TO 900-TERMINATION.
290      029000     MOVE SPACE TO TBLS-MRGD.
291      029100     MOVE ≠FIRST≠ TO INDX-KEY.
292      029200     GO TO 100-READ-TABLE.
293      029300
294      029400 120-READ-TABLE.
295      029500     MOVE TABL-NEXT TO INDX-KEY.
296      029600     PERFORM 700-READ-POINTER THRU 749-READ-POINTER-END.
297      029700     IF TABL-RFBY = 0
298      029800         GO TO 110-READ-TABLE.
299      029900     MOVE TABL-RFBY TO TABL-KEY.
300      030000     PERFORM 800-READ-TABLE THRU 849-READ-TABLE-END.
301      030100     IF EROR-CODE = 2
302      030200         GO TO 110-READ-TABLE.
303      030300     MOVE NTRY-DATA TO RFBY-RECD.
304      030400     IF RFBY-NUMB IS GREATER THAN 1
305      030500         GO TO 110-READ-TABLE.
306      030600     IF RFBY-TABL (1) = ≠FIRST≠
307      030700         GO TO 110-READ-TABLE.
308      030800     MOVE TABL-NEXT TO NEXT-TABL.
309      030900     MOVE TABL-NMBR TO MRGD-INDX.
310      031000     MOVE RFBY-TABL (1) TO RCVG-INDX.
311      031100     IF TABL-CNDS = 0
312      031200         MOVE ≠1≠ TO MRGE-FLAG
313      031300     ELSE
314      031400         MOVE ≠0≠ TO MRGE-FLAG.
```

```
315      031500      NOTE *****  MRGE-FLAG IS 1 FOR AN ACTION TABLE BEING MERGED.
316      031600      PERFORM 500-CHECK-REFERENCE THRU 599-CHECK-REFERENCE-END.
317      031700      IF TABL-CNDS = 0 AND
318      031800          MRGE-FLAG = ≠0≠
319      031900          MOVE ≠NO ≠ TO MRGE-TEST.
320      032000      IF MRGE-TEST NOT = ≠YES≠
321      032100          GO TO 140-SKIP-MERGE.
322      032200      PERFORM 200-COMBINE THRU 299-COMBINE-END.
323      032300
324      032400  140-SKIP-MERGE.
325      032500      MOVE NEXT-TABL TO TABL-NEXT.
326      032600      GO TO 110-READ-TABLE.
327      032700
328      032800  200-COMBINE.
329      032900      MOVE DTBL-PNTR TO RCVG-PNTR.
330      033000      MOVE MRGD-TABL TO TAB1-NAME.
331      033100      MOVE RCVG-NUMB TO TAB2-NAME.
332      033200      MOVE TABN-AREA TO ERDR-VALU .
333      033300      MOVE 8 TO ERDR-CODE.
334      033400      PERFORM 600-ERROR-ROUTINE THRU 699-ERROR-ROUTINE-END.
335      033500      COMPUTE RCVG-ITMS = RCVG-ITMS - RCVG-ACTS - RCVG-CNDS.
336      033600      MOVE MRGD-INDX TO INDX-KEY.
337      033700      PERFORM 700-READ-POINTER THRU 749-READ-POINTER-END.
338      033800      IF RCVG-RULS + TABL-RULS * NUMB-GOTO IS GREATER THAN 40
339      033900          GO TO 299-COMBINE-END.
340      034000      MOVE ≠X≠ TO TBLS-MRGD.
341      034100      SUBTRACT 1 FROM AVAL-TBLS.
342      034200      MOVE +0 TO INDX-GP01.
343      034300      MOVE RCVG-COND TO TABL-KEY.
344      034400      PERFORM 270-READ-TABLES THRU 279-READ-TABLES-END.
345      034500      MOVE INDX-GP01 TO CNDS-RCVG.
346      034600      MOVE TABL-COND TO TABL-KEY.
347      034700      PERFORM 270-READ-TABLES THRU 279-READ-TABLES-END.
348      034800      COMPUTE CNDS-MRGD = INDX-GP01 - CNDS-RCVG.
349      034900      MOVE RCVG-ACTN TO TABL-KEY.
350      035000      MOVE 1 TO CHCK-REQD.
351      035100      PERFORM 270-READ-TABLES THRU 279-READ-TABLES-END.
352      035200      MOVE 0 TO CHCK-REQD.
353      035300      COMPUTE ACTS-RCVG = INDX-GP01 - CNDS-RCVG - CNDS-MRGD.
354      035400      MOVE TABL-ACTN TO TABL-KEY.
355      035500      PERFORM 270-READ-TABLES THRU 279-READ-TABLES-END.
356      035600      COMPUTE ACTS-MRGD = INDX-GP01 - CNDS-RCVG - CNDS-MRGD
357      035700          - ACTS-RCVG.
358      035800      MOVE +0 TO INDX-GP01.
359      035900      PERFORM 260-INIT THRU 269-INIT-END.
360      036000      MOVE +0 TO INDX-GP01.
361      036100      MOVE RCVG-RULS TO TOTL-RULS.
362      036200      PERFORM 400-SHIFT THRU 499-SHIFT-END.
363      036300      MOVE TOTL-RULS TO RCVG-RULS.
364      036400      MOVE +0 TO INDX-GP01.
365      036500      PERFORM 300-MERGE THRU 399-MERGE-END.
366      036600      MOVE +0 TO INDX-GP01.
367      036700      MOVE AVAL-TERM TO LAST-INDX.
368      036800
369      036900  210-RELEASE.
370      037000      ADD +1 TO INDX-GP01.
```

```
371     037100     IF INDX-GP01 IS GREATER THAN NUMB-GOTO
372     037200         GO TO 220-RELEASE-REST.
373     037300     MOVE GOTO-RECD (INDX-GP01) TO NEXT-INDX.
374     037400     MOVE ≠X≠ TO RLSE-SNGL.
375     037500     PERFORM 280-CHAIN THRU 289-CHAIN-END.
376     037600     MOVE SPACE TO RLSE-SNGL.
377     037700     GO TO 210-RELEASE.
378     037800
379     037900 220-RELEASE-REST.
380     038000     MOVE TABL-INDX TO NEXT-INDX.
381     038100     PERFORM 280-CHAIN THRU 289-CHAIN-END.
382     038200     IF TABL-RFTO NOT = 0
383     038300         MOVE TABL-RFTO TO NEXT-INDX
384     038400         PERFORM 280-CHAIN THRU 289-CHAIN-END.
385     038500     IF TABL-RFBY NOT = 0
386     038600         MOVE TABL-RFBY TO NEXT-INDX
387     038700         PERFORM 280-CHAIN THRU 289-CHAIN-END.
388     038800     MOVE TOTL-RULS TO RCVG-RULS.
389     038900     COMPUTE RCVG-CNDS = CNDS-RCVG + CNDS-MRGD.
390     039000     COMPUTE RCVG-ACTS = ACTS-RCVG + ACTS-MRGD.
391     039100     COMPUTE RCVG-ITMS = RCVG-ITMS + RCVG-ACTS + RCVG-CNDS.
392     039200     MOVE LAST-INDX TO AVAL-TERM.
393     039300     PERFORM 250-DELETE-PNTR THRU 259-DELETE-PNTR-END.
394     039400     GO TO 299-COMBINE-END.
395     039500
396     039600 250-DELETE-PNTR.
397     039700     MOVE TABL-NFXT TO SAVE-NEXT.
398     039800     MOVE TABL-NFXT TO INDX-KEY.
399     039900     MOVE TABL-LAST TO SAVE-PREV.
400     040000     IF SAVE-NEXT = ≠AVAIL≠
401     040100         MOVE SAVE-PREV TO AVAL-LAST.
402     040200     IF SAVE-NEXT = 0
403     040300         GO TO 255-PREV-POINTER.
404     040400     PERFORM 700-READ-POINTER THRU 749-READ-POINTER-END.
405     040500     MOVE SAVE-PREV TO TABL-LAST.
406     040600     PERFORM 750-WRITE-POINTER THRU 799-WRITE-POINTER-END.
407     040700
408     040800 255-PREV-POINTER.
409     040900     IF SAVE-PREV NOT = RCVG-NMBR
410     041000         MOVE SAVE-PREV TO INDX-KEY
411     041100         PERFORM 700-READ-POINTER THRU 749-READ-POINTER-END
412     041200         MOVE SAVE-NEXT TO TABL-NFXT
413     041300         PERFORM 750-WRITE-POINTER THRU 799-WRITE-POINTER-END
414     041400     ELSE
415     041500         MOVE SAVE-NEXT TO RCVG-NFXT.
416     041600     MOVE RCVG-NMBR TO INDX-KEY.
417     041700     MOVE RCVG-PNTR TO DTBL-PNTR.
418     041800     PERFORM 750-WRITE-POINTER THRU 799-WRITE-POINTER-END.
419     041900
420     042000 259-DELETE-PNTR-END.
421     042100     EXIT.
422     042200
423     042300 260-INIT.
424     042400     ADD +1 TO INDX-GP01.
425     042500     IF INDX-GP01 IS GREATER THAN +40
426     042600         MOVE +0 TO NMBR-STBS
```

```
427      042700          GO TO 262-MOVE-STUBS.
428      042800      MOVE INDX-GP01 TO POSN-NIEW (INDX-GP01).
429      042900      GO TO 260-INIT.
430      043000
431      043100 262-MOVE-STUBS.
432      043200      MOVE CNDS-RCVG TO INDX-GP02.
433      043300      PERFORM 265-MOVE THRU 268-MOVE-END CNDS-MRGD TIMES.
434      043400      ADD ACTS-RCVG TO INDX-GP02.
435      043500      PERFORM 265-MOVE THRU 268-MOVE-END ACTS-MRGD TIMES.
436      043600      GO TO 269-INIT-END.
437      043700
438      043800 265-MOVE.
439      043900      ADD +1 TO INDX-GP02.
440      044000      MOVE DCSN-ITMS (INDX-GP02) TO TEMP-AREA.
441      044100      ADD +1 TO NMBR-STBS.
442      044200      MOVE TEMP-RULS TO MRGD-STBS (NMBR-STBS).
443      044300      MOVE SPACE TO TEMP-RULS.
444      044400      MOVE TEMP-AREA TO DCSN-ITMS (INDX-GP02).
445      044500
446      044600 268-MOVE-END.
447      044700      EXIT.
448      044800
449      044900 269-INIT-END.
450      045000      EXIT.
451      045100
452      045200 270-READ-TABLES.
453      045300      IF TABL-KEY = 0
454      045400          GO TO 279-READ-TABLES-END.
455      045500      PERFORM 800-READ-TABLE THRU 849-READ-TABLE-END.
456      045600      IF EROR-CODE NOT = 0
457      045700          MOVE 0 TO EROR-CODE
458      045800          GO TO 279-READ-TABLES-END.
459      045900      IF CHCK-REQD = 0
460      046000          GO TO 275-ENTER-IN-TABLE.
461      046100      MOVE +0 TO INDX-GP02.
462      046200
463      046300 273-SEARCH.
464      046400      ADD +1 TO INDX-GP02.
465      046500      IF INDX-GP02 IS GREATER THAN NUMB-GOTO
466      046600          GO TO 275-ENTER-IN-TABLE.
467      046700      IF GOTO-RECD (INDX-GP02) NOT = TABL-KEY
468      046800          GO TO 273-SEARCH.
469      046900      GO TO 277-NEXT-READ.
470      047000
471      047100 275-ENTER-IN-TABLE.
472      047200      ADD +1 TO INDX-GP01.
473      047300      IF INDX-GP01 IS GREATER THAN 100
474      047400          MOVE 5 TO EROR-CODE
475      047500          MOVE ≠275≠ TO EROR-VALU
476      047600          PERFORM 600-ERROR-ROUTINE THRU 699-ERROR-ROUTINE-END.
477      047700      MOVE DTBL-NTRY TO DCSN-ITMS (INDX-GP01).
478      047800
479      047900 277-NEXT-READ.
480      048000      IF NTRY-NEXT NOT = 0
481      048100          MOVE NTRY-NEXT TO TABL-KEY
482      048200          GO TO 270-READ-TABLES.
```

```
0483     048300
0484     048400 279-READ-TABLES-END.
0485     048500     EXIT.
0486     048600
0487     048700 280-CHAIN.
0488     048800     MOVE LAST-INDX TO TABL-KEY.
0489     048900     PERFORM 800-READ-TABLE THRU 849-READ-TABLE-END.
0490     049000     MOVE NEXT-INDX TO NTRY-NEXT.
0491     049100     PERFORM 850-WRITE-TABLE THRU 899-WRITE-TABLE-END.
0492     049200     MOVE NEXT-INDX TO TABL-KEY.
0493     049300     PERFORM 800-READ-TABLE THRU 849-READ-TABLE-END.
0494     049400     MOVE LAST-INDX TO NTRY-PREV.
0495     049500     IF RLSE-SNGL NOT = SPACE
0496     049600         MOVE 0 TO NTRY-NEXT.
0497     049700     MOVE NTRY-NEXT TO NEXT-INDX.
0498     049800     PERFORM 850-WRITE-TABLE THRU 899-WRITE-TABLE-END.
0499     049900     IF NEXT-INDX NOT = 0
0500     050000         PERFORM 290-READ THRU 298-READ-END.
0501     050100     MOVE TABL-KEY TO LAST-INDX.
0502     050200
0503     050300 289-CHAIN-END.
0504     050400     EXIT.
0505     050500
0506     050600 290-READ.
0507     050700     MOVE NEXT-INDX TO TABL-KEY.
0508     050800
0509     050900 295-READ.
0510     051000     PERFORM 800-READ-TABLE THRU 849-READ-TABLE-END.
0511     051100     IF NTRY-NEXT NOT = 0
0512     051200         MOVE NTRY-NEXT TO TABL-KEY
0513     051300         GO TO 295-READ.
0514     051400
0515     051500 298-READ-END.
0516     051600     EXIT.
0517     051700
0518     051800 299-COMBINE-END.
0519     051900     EXIT.
0520     052000
0521     052100 300-MERGE.
0522     052200     MOVE DTBL-PNTR TO SAVE-PNTR.
0523     052300     MOVE +0 TO INDX-GP02.
0524     052400     MOVE +0 TO INDX-GP04.
0525     052500     MOVE  0 TO LAST-RCRD.
0526     052600     COMPUTE INDX-GP03 = CNDS-RCVG + CNDS-MRGD.
0527     052700     PERFORM 390-CHAIN THRU 398-CHAIN-END INDX-GP03 TIMES.
0528     052800     MOVE 0 TO LAST-RCRD.
0529     052900     MOVE +0 TO INDX-GP04.
0530     053000     COMPUTE INDX-GP03 = ACTS-RCVG + ACTS-MRGD.
0531     053100     PERFORM 390-CHAIN THRU 398-CHAIN-END INDX-GP03 TIMES.
0532     053200     COMPUTE INDX-GP03 = CNDS-RCVG + CNDS-MRGD + ACTS-RCVG +
0533     053300         ACTS-MRGD.
0534     053400     MOVE +0 TO INDX-GP02.
0535     053500     PERFORM 380-WRITE THRU 389-WRITE-END.
0536     053600     MOVE RCVG-RETO TO TABL-KEY.
0537     053700
0538     053800 310-READ-RETO.
```

```
053900     PERFORM 800-READ-TABLE THRU 849-READ-TABLE-END.
054000     MOVE NTRY-DATA TO RFTO-RECD.
054100     MOVE RFTO-NUMB TO INDX-GP04.
054200     MOVE +0 TO RFTO-LNTH.
054300     MOVE +0 TO INDX-GP05.
054400
054500 315-STORE.
054600     ADD +1 TO INDX-GP05.
054700     IF INDX-GP05 IS GREATER THAN 23
054800         GO TO 320-READ-AGAIN.
054900     IF RFTO-LNTH NOT LESS THAN INDX-GP04
055000         GO TO 325-READ-OTHER-TABLE.
055100     IF RFTO-TABL (INDX-GP05) = MRGD-INDX
055200         SUBTRACT 1 FROM INDX-GP04
055300         GO TO 315-STORE.
055400     ADD +1 TO RFTO-LNTH.
055500     MOVE RFTO-TABL (INDX-GP05) TO RFTO-ITEM (RFTO-LNTH).
055600     GO TO 315-STORE.
055700
055800 320-READ-AGAIN.
055900     IF NTRY-NEXT = 0
056000         GO TO 325-READ-OTHER-TABLE.
056100     MOVE NTRY-NEXT TO TABL-KEY.
056200     PERFORM 800-READ-TABLE THRU 849-READ-TABLE-END.
056300     MOVE NTRY-DATA TO RFTO-RECD.
056400     ADD RFTO-NUMB TO INDX-GP04.
056500     MOVE +0 TO INDX-GP05.
056600     GO TO 315-STORE.
056700
056800 325-READ-OTHER-TABLE.
056900     IF TABL-RFTO = 0
057000         MOVE ALL ≠0≠ TO RFTO-RECD
057100         GO TO 327-INIT.
057200     MOVE TABL-RFTO TO TABL-KEY.
057300     PERFORM 800-READ-TABLE THRU 849-READ-TABLE-END.
057400     MOVE NTRY-DATA TO RFTO-RECD.
057500     MOVE RFTO-NUMB TO INDX-GP04.
057600
057700 327-INIT.
057800     MOVE +0 TO INDX-GP05.
057900     MOVE +0 TO INDX-GP06.
058000
058100 330-STORE.
058200     ADD +1 TO INDX-GP05.
058300     IF INDX-GP05 IS GREATER THAN 23
058400         GO TO 335-READ-AGAIN.
058500     IF INDX-GP06 NOT LESS THAN INDX-GP04
058600         GO TO 340-WRITE.
058700     ADD +1 TO INDX-GP06.
058800     MOVE RFTO-TABL (INDX-GP05) TO TEST-TABL.
058900     PERFORM 370-SCAN THRU 379-SCAN-END.
059000     IF INDX-GP07 = 0
059100         GO TO 330-STORE.
059200
059300     ADD +1 TO RFTO-LNTH.
059400     MOVE RFTO-TABL (INDX-GP05) TO RFTO-ITEM (RFTO-LNTH).
```

```
1595     059500     GO TO 330-STORE.
1596     059600
1597     059700 335-READ-AGAIN.
1598     059800     IF NTRY-NEXT = 0
1599     059900         GO TO 340-WRITE.
1600     060000     MOVE NTRY-NEXT TO TABL-KEY.
1601     060100     PERFORM 800-READ-TABLE THRU 849-READ-TABLE-END.
1602     060200     MOVE NTRY-DATA TO RFTO-RECD.
1603     060300     ADD RFTO-NUMB TO INDX-GP04.
1604     060400     MOVE +0 TO INDX-GP05.
1605     060500     GO TO 330-STORE.
1606     060600
1607     060700 340-WRITE.
1608     060800     MOVE +0 TO INDX-GP05.
1609     060900     MOVE +0 TO INDX-GP06.
1610     061000     MOVE RFTO-LNTH TO INDX-GP08.
1611     061100     MOVE RCVG-RFTO TO TABL-KEY.
1612     061200     MOVE SPACE TO RFTO-INFO.
1613     061300
1614     061400 345-MOVE.
1615     061500     ADD +1 TO INDX-GP05.
1616     061600     IF INDX-GP05 IS GREATER THAN 23
1617     061700         SUBTRACT 23 FROM INDX-GP08
1618     061800         MOVE 23 TO RFTO-NUMB
1619     061900         PERFORM 360-WRITE-RFTO THRU 369-WRITE-RFTO-END
1620     062000         GO TO 345-MOVE.
1621     062100     ADD +1 TO INDX-GP06.
1622     062200     IF INDX-GP06 IS GREATER THAN RFTO-LNTH
1623     062300         MOVE INDX-GP08 TO RFTO-NUMB
1624     062400         PERFORM 360-WRITE-RFTO THRU 369-WRITE-RFTO-END
1625     062500         GO TO 347-MOVE-END.
1626     062600     MOVE RFTO-ITEM (INDX-GP06) TO RFTO-TABL (INDX-GP05).
1627     062700     GO TO 345-MOVE.
1628     062800
1629     062900 347-MOVE-END.
1630     063000     MOVE SAVE-PNTR TO DTBL-PNTR.
1631     063100     MOVE SAVE-KEY  TO INDX-KEY.
1632     063200     GO TO 399-MERGE-END.
1633     063300 360-WRITE-RFTO.
1634     063400     PERFORM 800-READ-TABLE THRU 849-READ-TABLE-END.
1635     063500     MOVE RFTO-RECD TO NTRY-DATA.
1636     063600     MOVE SPACE TO RFTO-RECD.
1637     063700     IF INDX-GP06 NOT LESS THAN RFTO-LNTH
1638     063800         GO TO 362-WRITE-IT.
1639     063900     IF NTRY-NEXT NOT = 0
1640     064000         GO TO 362-WRITE-IT.
1641     064100     MOVE AVAL-RECD TO NTRY-NEXT.
1642     064200     MOVE NTRY-INDX TO LAST-INDX.
1643     064300     PERFORM 850-WRITE-TABLE THRU 899-WRITE-TABLE-END.
1644     064400     MOVE AVAL-RECD TO TABL-KEY.
1645     064500     PERFORM 800-READ-TABLE THRU 849-READ-TABLE-END.
1646     064600     MOVE LAST-INDX TO NTRY-PREV.
1647     064700     MOVE NTRY-NEXT TO NEXT-INDX.
1648     064800     MOVE 0 TO NTRY-NEXT.
1649     064900     PERFORM 850-WRITE-TABLE THRU 899-WRITE-TABLE-END.
1650     065000     MOVE NEXT-INDX TO AVAL-RECD TABL-KEY.
```

```
651      065100      PERFORM 800-READ-TABLE THRU 849-READ-TABLE-END.
652      065200      MOVE 0 TO NTRY-PREV.
653      065300      PERFORM 850-WRITE-TABLE THRU 899-WRITE-TABLE-END.
654      065400      GO TO 368-WRITE-RFTO-END.
655      065500
656      065600  362-WRITE-IT.
657      065700      PERFORM 850-WRITE-TABLE THRU 899-WRITE-TABLE-END.
658      065800      MOVE NTRY-NEXT TO TABL-KEY.
659      065900
660      066000  368-WRITE-RFTO-END.
661      066100      MOVE +0 TO INDX-GP05.
662      066200
663      066300  369-WRITE-RFTO-END.
664      066400      EXIT.
665      066500
666      066600  370-SCAN.
667      066700      MOVE TEST-TABL TO INDX-KEY.
668      066800      PERFORM 700-READ-POINTER THRU 749-READ-POINTER-END.
669      066900      MOVE TABL-RFBY TO TABL-KEY.
670      067000      MOVE +0 TO RFBY-LNTH.
671      067100
672      067200  371-SCAN.
673      067300      PERFORM 800-READ-TABLE THRU 849-READ-TABLE-END.
674      067400      MOVE NTRY-DATA TO RFBY-RECD.
675      067500      MOVE +0 TO INDX-GP07.
676      067600
677      067700  372-CHECK-RCVG.
678      067800      ADD +1 TO INDX-GP07.
679      067900      IF INDX-GP07 IS GREATER THAN RFBY-NUMB
680      068000          GO TO 373-GET-NEXT-RECORD.
681      068100      IF RFBY-TABL (INDX-GP07) = RCVG-NMBR
682      068200          GO TO 372-CHECK-RCVG.
683      068300      ADD +1 TO RFBY-LNTH.
684      068400      IF RFBY-TABL (INDX-GP07) = MRGD-INDX
685      068500          MOVE RCVG-NMBR TO RFBY-ITEM (RFBY-LNTH)
686      068600      ELSE
687      068700          MOVE RFBY-TABL (INDX-GP07) TO RFBY-ITEM (RFBY-LNTH).
688      068800      GO TO 372-CHECK-RCVG.
689      068900
690      069000  373-GET-NEXT-RECORD.
691      069100      IF NTRY-NEXT = 0
692      069200          GO TO 374-WRITE.
693      069300      MOVE NTRY-NEXT TO TABL-KEY.
694      069400      GO TO 371-SCAN.
695      069500
696      069600  374-WRITE.
697      069700      MOVE +0 TO INDX-GP07.
698      069800      MOVE TABL-RFBY TO TABL-KEY.
699      069900
700      070000  375-READ-IT-IN.
701      070100      MOVE SPACE TO RFBY-RECD.
702      070200      PERFORM 800-READ-TABLE THRU 849-READ-TABLE-END.
703      070300      MOVE +0 TO INDX-GP08.
704      070400
705      070500  376-MOVE.
706      070600      ADD +1 TO INDX-GP08.
```

```
070700      IF INDX-GP08 IS GREATER THAN 23
070800          MOVE 23 TO RFBY-NUMB
070900          MOVE RFBY-RECD TO NTRY-DATA
071000          PERFORM 850-WRITE-TABLE THRU 899-WRITE-TABLE-END
071100          MOVE NTRY-NEXT TO TABL-KEY
071200          GO TO 375-READ-IT-IN.
071300      ADD +1 TO INDX-GP07.
071400      MOVE RFBY-ITEM (INDX-GP07) TO RFBY-TABL (INDX-GP08).
071500      IF INDX-GP07 IS LESS THAN RFBY-LNTH
071600          GO TO 376-MOVE.
071700      MOVE +0 TO INDX-GP07.
071800      MOVE INDX-GP08 TO RFBY-NUMB.
071900      MOVE RFBY-RECD TO NTRY-DATA.
072000      MOVE NTRY-NEXT TO NEXT-INDX.
072100      MOVE 0 TO NTRY-NEXT.
072200      PERFORM 850-WRITE-TABLE THRU 899-WRITE-TABLE-END.
072300      IF NEXT-INDX = 0
072400          GO TO 378-CHECK-RFTO.
072500      MOVE AVAL-TERM TO LAST-INDX.
072600      PERFORM 280-CHAIN THRU 289-CHAIN-END.
072700      MOVE LAST-INDX TO AVAL-TERM.
072800
072900 378-CHECK-RFTO.
073000      ADD +1 TO INDX-GP07.
073100      IF INDX-GP07 IS GREATER THAN RFTO-LNTH
073200          GO TO 379-SCAN-END.
073300      IF RFTO-ITEM (INDX-GP07) = TEST-TABL
073400          MOVE +0 TO INDX-GP07
073500          GO TO 379-SCAN-END.
073600      GO TO 378-CHECK-RFTO.
073700
073800 379-SCAN-END.
073900      EXIT.
074000
074100 380-WRITE.
074200      MOVE ITMS-CRNT (1) TO RCVG-ACTN.
074300      ADD +1 TO INDX-GP02.
074400
074500 382-WRITE.
074600      MOVE DCSN-ITMS (INDX-GP02) TO DTBL-NTRY.
074700      MOVE NTRY-INDX TO TABL-KEY.
074800      PERFORM 850-WRITE-TABLE THRU 899-WRITE-TABLE-END.
074900      IF INDX-GP02 NOT LESS THAN INDX-GP03
075000          GO TO 389-WRITE-END.
075100      ADD +1 TO INDX-GP02.
075200      IF NTRY-NEXT = 0
075300          MOVE RCVG-ACTN TO RCVG-COND
075400          MOVE ITMS-CRNT (INDX-GP02) TO RCVG-ACTN.
075500      GO TO 382-WRITE.
075600
075700 389-WRITE-END.
075800      EXIT.
075900
076000 390-CHAIN.
076100      ADD +1 TO INDX-GP04.
076200      ADD +1 TO INDX-GP02.
```

```
763     076300      MOVE LAST-RCRD TO ITMS-PREV (INDX-GP02).
764     076400      MOVE ITMS-CRNT (INDX-GP02) TO LAST-RCRD.
765     076500      IF INDX-GP04 NOT LESS THAN INDX-GP03
766     076600          MOVE 0 TO ITMS-NEXT (INDX-GP02)
767     076700          GO TO 398-CHAIN-END.
768     076800      COMPUTE INDX-GP05 = INDX-GP02 + 1.
769     076900      MOVE ITMS-CRNT (INDX-GP05) TO ITMS-NEXT (INDX-GP02).
770     077000
771     077100 398-CHAIN-END.
772     077200      EXIT.
773     077300
774     077400 399-MERGE-END.
775     077500      EXIT.
776     077600
777     077700 400-SHIFT.
778     077800      ADD +1 TO INDX-GP01.
779     077900      IF INDX-GP01 IS GREATER THAN NUMB-GOTO
780     078000          GO TO 499-SHIFT-END.
781     078100      MOVE GOTO-RECD (INDX-GP01) TO TABL-KEY.
782     078200      PERFORM 800-READ-TABLE THRU 849-READ-TABLE-END.
783     078300      IF EROR-CODE NOT = 0
784     078400          MOVE 0 TO EROR-CODE
785     078500          GO TO 400-SHIFT.
786     078600      MOVE NTRY-DATA TO ACTN-RECD.
787     078700      MOVE +0 TO INDX-GP02.
788     078800
789     078900 410-SEARCH.
790     079000      ADD +1 TO INDX-GP02.
791     079100      IF INDX-GP02 IS GREATER THAN RCVG-RULS
792     079200          GO TO 400-SHIFT.
793     079300      IF ACTN-RULE (INDX-GP02) = SPACE
794     079400          GO TO 410-SEARCH.
795     079500      MOVE INDX-GP02 TO INDX-GP05.
796     079600      MOVE POSN-NIEW (INDX-GP05) TO INDX-GP06.
797     079700      COMPUTE INDX-GP03 = TOTL-RULS - INDX-GP06 + 1.
798     079800      COMPUTE INDX-GP04 = TABL-RULS - 1.
799     079900      COMPUTE INDX-GP10 = INDX-GP03 + INDX-GP04.
800     080000      IF INDX-GP04 IS GREATER THAN +0
801     080100          PERFORM 480-SHIFT-RULES THRU 498-SHIFT-RULES-END.
802     080200      PERFORM 460-INSERT THRU 479-INSERT-END.
803     080300      GO TO 410-SEARCH.
804     080400
805     080500 460-INSERT.
806     080600      MOVE CNDS-RCVG TO INDX-GP05.
807     080700      MOVE +0 TO INDX-GP07.
808     080800      PERFORM 465-DO-ENTRY THRU 478-DO-ENTRY-END  CNDS-MRGD TIMES.
809     080900      COMPUTE INDX-GP05 = CNDS-RCVG + CNDS-MRGD + ACTS-RCVG.
810     081000      PERFORM 465-DO-ENTRY THRU 478-DO-ENTRY-END  ACTS-MRGD TIMES.
811     081100      GO TO 479-INSERT-END.
812     081200
813     081300 465-DO-ENTRY.
814     081400      ADD +1 TO INDX-GP05.
815     081500      MOVE DCSN-ITMS (INDX-GP05) TO TEMP-AREA.
816     081600      ADD +1 TO INDX-GP07.
817     081700      MOVE MRGD-STRS (INDX-GP07) TO SHFT-AREA.
818     081800      COMPUTE INDX-GP08 = INDX-GP06 - 1.
```

```
819      081900      MOVE +0 TO INDX-GP09.
820      082000      PERFORM 470-MOVE-EM THRU 477-MOVE-EM-END TABL-RULS TIMES.
821      082100      MOVE TEMP-AREA TO DCSN-ITMS (INDX-GP05).
822      082200      GO TO 478-DO-ENTRY-END.
823      082300
824      082400 470-MOVE-EM.
825      082500      ADD +1 TO INDX-GP08.
826      082600      ADD +1 TO INDX-GP09.
827      082700      MOVE SHFT-RULE (INDX-GP09) TO TEMP-RULE (INDX-GP08).
828      082800
829      082900 477-MOVE-EM-END.
830      083000      EXIT.
831      083100
832      083200 478-DO-ENTRY-END.
833      083300      EXIT.
834      083400
835      083500 479-INSERT-END.
836      083600      EXIT.
837      083700
838      083800 480-SHIFT-RULES.
839      083900      ADD INDX-GP04 TO TOTL-RULS.
840      084000      COMPUTE INDX-GP11 = CNDS-RCVG + ACTS-RCVG + CNDS-MRGD +
841      084100          ACTS-MRGD.
842      084200      MOVE +0 TO INDX-GP07.
843      084300      PERFORM 485-SHIFT-RULES THRU 497-SHIFT-RULES-END INDX-GP11
844      084400          TIMES.
845      084500
846      084600 482-UPDATE-POSN-TABL.
847      084700      ADD +1 TO INDX-GP05.
848      084800      IF INDX-GP05 IS GREATER THAN 40
849      084900          GO TO 498-SHIFT-RULES-END.
850      085000      ADD INDX-GP04 TO POSN-NIEW (INDX-GP05).
851      085100      GO TO 482-UPDATE-POSN-TABL.
852      085200
853      085300 485-SHIFT-RULES.
854      085400      ADD +1 TO INDX-GP07.
855      085500      MOVE DCSN-ITMS (INDX-GP07) TO TEMP-AREA.
856      085600      MOVE SPACE TO SHFT-AREA.
857      085700      COMPUTE INDX-GP08 = INDX-GP06 - 1.
858      085800      MOVE +0 TO INDX-GP09.
859      085900
860      086000 488-RULE-TO-SHIFT.
861      086100      ADD +1 TO INDX-GP09.
862      086200      IF INDX-GP09 IS GREATER THAN INDX-GP03
863      086300          GO TO 490-GO-RIGHT.
864      086400      ADD +1 TO INDX-GP08.
865      086500      MOVE TEMP-RULE (INDX-GP08) TO SHFT-RULE (INDX-GP09).
866      086600      GO TO 488-RULE-TO-SHIFT.
867      086700
868      086800 490-GO-RIGHT.
869      086900      MOVE +0 TO INDX-GP09.
870      087000
871      087100 492-GO-RIGHT.
872      087200      ADD +1 TO INDX-GP09.
873      087300      IF INDX-GP09 IS GREATER THAN INDX-GP04
874      087400          COMPUTE INDX-GP08 = INDX-GP06 - 1
```

```
875      087500          MOVE +0 TO INDX-GP09
876      087600          GO TO 495-SHIFT-TO-RULE.
877      087700      MOVE SHFT-WHOL TO SHFT-REST.
878      087800      GO TO 492-GO-RIGHT.
879      087900
880      088000 495-SHIFT-TO-RULE.
881      088100      ADD +1 TO INDX-GP09.
882      088200      IF INDX-GP09 IS GREATER THAN INDX-GP10
883      088300          GO TO 496-PUT-IT-BACK.
884      088400      ADD +1 TO INDX-GP08.
885      088500      MOVE SHFT-RULE (INDX-GP09) TO TEMP-RULE (INDX-GP08).
886      088600      GO TO 495-SHIFT-TO-RULE.
887      088700
888      088800 496-PUT-IT-BACK.
889      088900      MOVE TEMP-AREA TO DCSN-ITMS (INDX-GP07).
890      089000
891      089100 497-SHIFT-RULES-END.
892      089200      EXIT.
893      089300
894      089400 498-SHIFT-RULES-END.
895      089500      EXIT.
896      089600
897      089700 499-SHIFT-END.
898      089800      EXIT.
899      089900
900      090000 500-CHECK-REFERENCE.
901      090100      MOVE +0 TO NUMB-GOTO.
902      090200      MOVE RCVG-INDX TO INDX-KEY.
903      090300      PERFORM 700-READ-POINTER THRU 749-READ-POINTER-END.
904      090400      MOVE SPACE TO RULE-FLGS.
905      090500      MOVE TABL-ACTN TO TABL-KEY.
906      090600      MOVE ≠YES≠ TO MRGE-TEST.
907      090700
908      090800 510-READ-ACTIONS.
909      090900      PERFORM 800-READ-TABLE THRU 849-READ-TABLE-END.
910      091000      IF EROR-CODE = 2
911      091100          MOVE ≠NO ≠ TO MRGE-TEST
912      091200          GO TO 599-CHECK-REFERENCE-END.
913      091300      MOVE NTRY-DATA TO ACTN-RECD.
914      091400      IF GOTO-CNST NOT = ≠GO TO ≠
915      091500          GO TO 520-NOT-GO.
916      091600      IF GOTO-NAME NOT = MRGD-TABL
917      091700          GO TO 530-GET-NEXT-ACTN.
918      091800      IF MRGE-FLAG = 1
919      091900          GO TO 515-STORE-GO.
920      092000      IF TABL-CNDS = 0
921      092100          GO TO 515-STORE-GO.
922      092200      MOVE +0 TO INDX-GP01.
923      092300      PERFORM 590-VERIFY-MRG THRU 598-VERIFY-MRG-END TABL-RULS
924      092400          TIMES.
925      092500
926      092600 515-STORE-GO.
927      092700      IF MRGE-TEST NOT = ≠YES≠
928      092800          GO TO 599-CHECK-REFERENCE-END.
929      092900      ADD +1 TO NUMB-GOTO.
930      093000      IF NUMB-GOTO IS GREATER THAN 10
```

```
931      093100          MOVE ≠NO ≠ TO MRGE-TEST
932      093200          GO TO 540-DUPLICATE-GOTO.
933      093300      MOVE NTRY-INDX TO GOTO-RECD (NUMB-GOTO).
934      093400      GO TO 530-GET-NEXT-ACTN.
935      093500
936      093600 520-NOT-GO.
937      093700      IF TABL-CNDS = 0
938      093800          GO TO 530-GET-NEXT-ACTN.
939      093900      MOVE +0 TO INDX-GP01.
940      094000      PERFORM 580-CHECK-RULES THRU 589-CHECK-RULES-END TABL-RULS
941      094100          TIMES.
942      094200
943      094300 530-GET-NEXT-ACTN.
944      094400      IF NTRY-NEXT = 0
945      094500          GO TO 540-DUPLICATE-GOTO.
946      094600      MOVE NTRY-NEXT TO TABL-KEY.
947      094700      GO TO 510-READ-ACTIONS.
948      094800
949      094900 540-DUPLICATE-GOTO.
950      095000      IF NUMB-GOTO = 1
951      095100          GO TO 599-CHECK-REFERENCE-END.
952      095200      IF NUMB-GOTO IS GREATER THAN 10
953      095300          MOVE 10 TO NUMB-GOTO.
954      095400      MOVE GOTO-RECD (NUMB-GOTO) TO TABL-KEY.
955      095500      PERFORM 800-READ-TABLE THRU 849-READ-TABLE-END.
956      095600      MOVE NTRY-DATA TO ACTN-RECD.
957      095700      MOVE NTRY-INDX TO GOTO-THIS.
958      095800      MOVE NTRY-PREV TO GOTO-PREV.
959      095900      MOVE NTRY-NEXT TO GOTO-NEXT.
960      096000      MOVE +0 TO INDX-GP01.
961      096100
962      096200 545-GET-OTHER-GOTO.
963      096300      ADD +1 TO INDX-GP01.
964      096400      IF INDX-GP01 = NUMB-GOTO
965      096500          MOVE GOTO-RECD (NUMB-GOTO) TO GOTO-RECD (1)
966      096600          MOVE 1 TO NUMB-GOTO
967      096700          MOVE GOTO-THIS TO TABL-KEY NTRY-INDX
968      096800          MOVE GOTO-PREV TO NTRY-PREV
969      096900          MOVE GOTO-NEXT TO NTRY-NEXT
970      097000          MOVE ACTN-RECD TO NTRY-DATA
971      097100          PERFORM 750-WRITE-POINTER THRU 799-WRITE-POINTER-END
972      097200          PERFORM 850-WRITE-TABLE THRU 899-WRITE-TABLE-END
973      097300          GO TO 599-CHECK-REFERENCE-END.
974      097400      MOVE GOTO-RECD (INDX-GP01) TO TABL-KEY.
975      097500      PERFORM 800-READ-TABLE THRU 849-READ-TABLE-END.
976      097600      MOVE NTRY-DATA TO GOTO-LINE.
977      097700      MOVE +0 TO INDX-GP02.
978      097800      PERFORM 570-COMBINE-RULES THRU 579-COMBINE-RULES-END.
979      097900      MOVE NTRY-PREV TO LAST-INDX.
980      098000      MOVE NTRY-NEXT TO NEXT-INDX.
981      098100      IF LAST-INDX = 0
982      098200          MOVE NEXT-INDX TO TABL-ACTN
983      098300          GO TO 550-CHAIN.
984      098400      MOVE LAST-INDX TO TABL-KEY.
985      098500      PERFORM 800-READ-TABLE THRU 849-READ-TABLE-END.
986      098600      MOVE NEXT-INDX TO NTRY-NEXT.
```

```
0987     098700     PERFORM 850-WRITE-TABLE THRU 899-WRITE-TABLE-END.
0988     098800
0989     098900 550-CHAIN.
0990     099000     SUBTRACT 1 FROM TABL-ACTS.
0991     099100     SUBTRACT 1 FROM TABL-ITMS.
0992     099200     MOVE NEXT-INDX TO TABL-KEY.
0993     099300     PERFORM 800-READ-TABLE THRU 849-READ-TABLE-END.
0994     099400     MOVE LAST-INDX TO NTRY-PREV.
0995     099500     PERFORM 850-WRITE-TABLE THRU 899-WRITE-TABLE-END.
0996     099600     MOVE "X" TO RLSE-SNGL.
0997     099700     MOVE AVAL-TERM TO LAST-INDX.
0998     099800     MOVE GOTO-RECD (INDX-GP01) TO NEXT-INDX.
0999     099900     PERFORM 280-CHAIN THRU 289-CHAIN-END.
1000     100000     MOVE SPACE TO RLSE-SNGL.
1001     100100     GO TO 545-GET-OTHER-GOTO.
1002     100200
1003     100300 570-COMBINE-RULES.
1004     100400     ADD +1 TO INDX-GP02.
1005     100500     IF INDX-GP02 IS GREATER THAN 40
1006     100600         GO TO 579-COMBINE-RULES-END.
1007     100700     IF GOTO-RULE (INDX-GP02) NOT = SPACE
1008     100800         MOVE "X" TO ACTN-RULE (INDX-GP02).
1009     100900     GO TO 570-COMBINE-RULES.
1010     101000
1011     101100 579-COMBINE-RULES-END.
1012     101200     EXIT.
1013     101300
1014     101400 580-CHECK-RULES.
1015     101500     ADD +1 TO INDX-GP01.
1016     101600     IF ACTN-RULE (INDX-GP01) = SPACE
1017     101700         GO TO 589-CHECK-RULES-END.
1018     101800     MOVE "X" TO RULE-FLAG (INDX-GP01).
1019     101900
1020     102000 589-CHECK-RULES-END.
1021     102100     EXIT.
1022     102200
1023     102300 590-VERIFY-MRG.
1024     102400     ADD +1 TO INDX-GP01.
1025     102500     IF ACTN-RULE (INDX-GP01) = SPACE
1026     102600         GO TO 598-VERIFY-MRG-END.
1027     102700     IF RULE-FLAG (INDX-GP01) NOT = SPACE
1028     102800         MOVE "NO " TO MRGE-TEST.
1029     102900
1030     103000 598-VERIFY-MRG-END.
1031     103100     EXIT.
1032     103200
1033     103300 599-CHECK-REFERENCE-END.
1034     103400     EXIT.
1035     103500
1036     103600 600-ERROR-ROUTINE.
1037     103700     MOVE EROR-CODE TO MESG-NMBR.
1038     103800     MOVE ERROR-MESG (ERROR-CODE) TO ERROR-TEXT.
1039     103900     DISPLAY ERROR-AREA.
1040     104000     MOVE SPACE TO ERROR-TEXT ERROR-VALU.
1041     104100     IF ERROR-CODE = 1 OR 4 OR 5
1042     104200         GO TO 950-ERROR-TERM.
```

```
1043     104300
1044     104400 699-ERROR-ROUTINE-END.
1045     104500     EXIT.
1046     104600
1047     104700 700-READ-POINTER.
1048     104800     READ DTINDEX INVALID KEY
1049     104900         MOVE 1 TO EROR-CODE
1050     105000         MOVE INDX-KEY TO EROR-VALU
1051     105100         PERFORM 600-ERROR-ROUTINE THRU 699-ERROR-ROUTINE-END
1052     105200         GO TO 749-READ-POINTER-END.
1053     105300     MOVE 0 TO EROR-CODE.
1054     105400
1055     105500 749-READ-POINTER-END.
1056     105600     EXIT.
1057     105700
1058     105800 750-WRITE-POINTER.
1059     105900     WRITE DTBL-PNTR INVALID KEY
1060     106000         MOVE 4 TO EROR-CODE
1061     106100         MOVE INDX-KEY TO EROR-VALU
1062     106200         PERFORM 600-ERROR-ROUTINE THRU 699-ERROR-ROUTINE-END
1063     106300         GO TO 799-WRITE-POINTER-END.
1064     106400     MOVE 0 TO EROR-CODE.
1065     106500
1066     106600 799-WRITE-POINTER-END.
1067     106700     EXIT.
1068     106800
1069     106900 800-READ-TABLE.
1070     107000     READ DETABLS INVALID KEY
1071     107100         MOVE 2 TO EROR-CODE
1072     107200         MOVE TABL-KEY TO EROR-VALU
1073     107300         MOVE 0 TO TABL-KEY
1074     107400         PERFORM 600-ERROR-ROUTINE THRU 699-ERROR-ROUTINE-END
1075     107500         GO TO 849-READ-TABLE-END.
1076     107600     MOVE 0 TO EROR-CODE.
1077     107700
1078     107800 849-READ-TABLE-END.
1079     107900     EXIT.
1080     108000
1081     108100 850-WRITE-TABLE.
1082     108200     WRITE DTBL-NTRY INVALID KEY
1083     108300         MOVE 3 TO EROR-CODE
1084     108400         MOVE TABL-KEY TO EROR-VALU
1085     108500         MOVE 0 TO TABL-KEY
1086     108600         PERFORM 600-ERROR-ROUTINE THRU 699-ERROR-ROUTINE-END
1087     108700         GO TO 899-WRITE-TABLE-END.
1088     108800     MOVE 0 TO EROR-CODE.
1089     108900
1090     109000 899-WRITE-TABLE-END.
1091     109100     EXIT.
1092     109200
1093     109300 900-TERMINATION.
1094     109400     IF NOOF-PSES IS GREATER THAN MAXM-PSES
1095     109500         GO TO 910-TERMINATION.
1096     109600     MOVE 7 TO EROR-CODE.
1097     109700     SUBTRACT 2 FROM NOOF-PSES.
1098     109800     MOVE NOOF-PSES TO EROR-VALU.
```

```
1099      109900     PERFORM 600-ERROR-ROUTINE THRU 699-ERROR-ROUTINE-END.
1100      110000
1101      110100 910-TERMINATION.
1102      110200     MOVE +0 TO RETN-CODE.
1103      110300     MOVE ≠AVAIL≠ TO INDX-KEY.
1104      110400     MOVE AVAL-PNTR TO DTBL-PNTR.
1105      110500     PERFORM 750-WRITE-POINTER THRU 799-WRITE-POINTER-END.
1106      110600     GO TO 999-ENDIT.
1107      110700
1108      110800 950-ERROR-TERM.
1109      110900     MOVE +99 TO RETN-CODE.
1110      111000
1111      111100 999-ENDIT.
1112      111200     CLOSE DTINDEX.
1113      111300     CLOSE DETABLS.
1114      111400     RETURN.
1115      111500
20 LENGTH IS  012022
00R SCM USED
```

```
0001     000100 IDENTIFICATION DIVISION.
0002     000200 PROGRAM-ID. DETBL30.
0003     000300 INSTALLATION. SIR GEORGE WILLIAMS UNIVERSITY.
0004     000400 DATE-COMPILED. MARCH 2, 75/07/18..
0005     000500 REMARKS.
0006     000600         THIS PROGRAM WILL PERFORM PRINT FUNCTIONS IN THE
0007     000700     DECISION TABLE GENERATING SYSTEM.
0008     000800
0009     000900 ENVIRONMENT DIVISION.
0010     001000 CONFIGURATION SECTION.
0011     001100 SOURCE-COMPUTER. 6400.
0012     001200 OBJECT-COMPUTER. 6400.
0013     001300 INPUT-OUTPUT SECTION.
0014     001400 FILE-CONTROL.
0015     001500     SELECT DTINDEX ASSIGN TO DTINDEX
0016     001600         ORGANIZATION IS STANDARD
0017     001700         FILE-LIMIT IS 1000
0018     001800         ACCESS MODE IS RANDOM
0019     001900         SYMBOLIC KEY IS INDX-KEY.
0020     002000
0021     002100     SELECT DETABLS ASSIGN TO DECTBLS
0022     002200         FILE-LIMIT IS 10000
0023     002300         ACCESS MODE IS RANDOM
0024     002400         ACTUAL KEY IS TABL-KEY.
0025     002500
0026     002600 DATA DIVISION.
0027     002700 FILE SECTION.
0028     002800
0029     002900 FD  DTINDEX
0030     003000     BLOCK CONTAINS 1 RECORDS
0031     003100     RECORD CONTAINS 60 CHARACTERS
0032     003200     DATA RECORD IS DTBL-PNTR
0033     003300     LABEL RECORDS ARE OMITTED.
0034     003400
0035     003500 01  DTBL-PNTR.
0036     003600     05  TABL-NMBR        PIC X(5).
0037     003700     05  TABL-NAME REDEFINES TABL-NMBR PIC 999V99.
0038     003800     05  TABL-NEXT        PIC X(5).
0039     003900     05  TABL-LAST        PIC X(5).
0040     004000     05  TABL-INDX        PIC 9(5).
0041     004100     05  TABL-RFTO        PIC 9(5).
0042     004200     05  TABL-RFBY        PIC 9(5).
0043     004300     05  TABL-COND        PIC 9(5).
0044     004400     05  TABL-ACTN        PIC 9(5).
0045     004500     05  TABL-ITMS        PIC 9(5).
0046     004600     05  TABL-ACTS        PIC 9(5).
0047     004700     05  TABL-CNDS        PIC 9(5).
0048     004800     05  TABL-RULS        PIC 9(5).
0049     004900
0050     005000 FD  DETABLS
0051     005100     BLOCK CONTAINS 1 RECORDS
0052     005200     RECORD CONTAINS 132 CHARACTERS
0053     005300     DATA RECORD IS DTBL-NTRY
0054     005400     LABEL RECORDS ARE OMITTED.
0055     005500
0056     005600 01  DTBL-NTRY.
```

```
0057     005700     05  NTRY-COMN.
0058     005800         10  NTRY-INDX    PIC 9(5).
0059     005900         10  NTRY-PREV    PIC 9(5).
0060     006000         10  NTRY-NEXT    PIC 9(5).
0061     006100     05  NTRY-DATA       PIC X(117).
0062     006200
0063     006300 COMMON-STORAGE SECTION.
0064     006400 01  COMN-INFO.
0065     006500     05  RETN-CODE       PIC S9(4) COMP.
0066     006600     05  PASD-INFO       PIC X(138).
0067     006700     05  REPT-LIST REDEFINES PASD-INFO.
0068     006800         10  RQST-CODE   PIC X            OCCURS 10 TIMES.
0069     006900         10  REPT-FL05   PIC X(128).
0070     007000
0071     007100 WORKING-STORAGE SECTION.
0072     007200 77  INDX-KEY            PIC X(5)         VALUE ≠FIRST≠.
0073     007300 77  TABL-KEY            PIC 9(5)         VALUE 0.
0074     007400 77  INDX-GP01           PIC 9(4) COMP    VALUE 0.
0075     007500 77  ERDR-CODE           PIC 9(4) COMP    VALUE 0.
0076     007600 77  TOTL-LNES           PIC 9(5)         VALUE 0.
0077     007700 77  LINE-CNTR           PIC 9(4) COMP    VALUE 75.
0078     007800 77  PAGE-CNTR           PIC 9(4) COMP    VALUE 0.
0079     007900 77  INDX-GP02           PIC 9(4) COMP    VALUE 0.
0080     008000 77  F-FLAG              PIC X            VALUE SPACE.
0081     008100 77  H-FLAG              PIC X            VALUE SPACE.
0082     008200
0083     008300 01  ERDR-MSGS.
0084     008400     05  ERDR-TABL.
0085     008500       10  MSG01 PIC X(30) VALUE ≠TABLE NOT FOUND - TERMINATE   ≠.
0086     008600       10  MSG02 PIC X(30) VALUE ≠NO ENTRY IN FILE - SKIP       ≠.
0087     008700       10  MSG03 PIC X(30) VALUE ≠DECISION TABLE PRINT REQUESTED≠.
0088     008800     05  ERDR-MESG REDEFINES ERDR-TABL OCCURS 3 TIMES PIC X(30).
0089     008900
0090     009000 01  PRNT-RECD.
0091     009100     05  PRNT-CODE       PIC X            VALUE ≠F≠.
0092     009200     05  PRNT-CCTL       PIC X            VALUE SPACE.
0093     009300     05  PRNT-DATA.
0094     009400         10  PRNT-POS1   PIC X            VALUE SPACE.
0095     009500         10  PRNT-FL05   PIC X(122)       VALUE SPACE.
0096     009600         10  PRNT-POSL   PIC X            VALUE SPACE.
0097     009700         10  PRNT-FL10   PIC X(8)         VALUE SPACE.
0098     009800
0099     009900 01  ERDR-AREA.
0100     010000     05  MESG-IDNT       PIC X(9)         VALUE ≠*** DET30≠.
0101     010100     05  MESG-NMBR       PIC 99           VALUE 0.
0102     010200     05  MESG-FL05       PIC X(5)         VALUE ≠ --- ≠.
0103     010300     05  ERDR-TEXT       PIC X(30)        VALUE SPACE.
0104     010400     05  ERDR-FL01       PIC X            VALUE SPACE.
0105     010500     05  ERDR-VALU       PIC X(84)        VALUE SPACE.
0106     010600
0107     010700 01  NAME-RECD.
0108     010800     05  NAME-NMBR       PIC 999V99.
0109     010900     05  NAME-ORIG       PIC X(106)       VALUE SPACE.
0110     011000
0111     011100 01  COND-RECD.
0112     011200     05  COND-INFO       PIC X(77)        VALUE SPACE.
```

```
011300      05  COND-FL01.
011400          10  COND-RULE   PIC X        OCCURS 40 TIMES.
011500
011600 01   ACTN-RECD.
011700      05  ACTN-INFO       PIC X(77)        VALUE SPACE.
011800      05  ACTN-GOTO REDEFINES ACTN-INFO.
011900          10  GOTO-CNST   PIC X(6).
012000          10  GOTO-NAME   PIC 999V99.
012100          10  GOTO-FL05   PIC X(66).
012200      05  ACTN-FL01.
012300          10  ACTN-RULE   PIC X       OCCURS 40 TIMES.
012400
012500 01   RFTO-RECD.
012600      05  RFTO-NUMB       PIC 99           VALUE 0.
012700      05  RFTO-ALFA.
012800          10  RFTO-TABX   PIC X(5) OCCURS 23 TIMES.
012900      05  RFTO-INFO REDEFINES RFTO-ALFA.
013000          10  RFTO-TABL   PIC 999V99 OCCURS 23 TIMES.
013100
013200 01   RFBY-RECD.
013300      05  RFBY-NUMB       PIC 99           VALUE 0.
013400      05  RFBY-ALFA.
013500          10  RFBY-TABX   PIC X(5) OCCURS 23 TIMES.
013600      05  RFBY-INFO REDEFINES RFBY-ALFA.
013700          10  RFBY-TABL   PIC 999V99 OCCURS 23 TIMES.
013800
013900 01   FXGO-LINE.
014000      05  FXGO-FL05       PIC X(6)         VALUE ≠GO TO ≠.
014100      05  FXGO-NAME       PIC 999.99.
014200
014300 01   PRNT-LNES.
014400      05  PRNT-LNE1.
014500          10  LNE1-FL01   PIC X(12)        VALUE ≠I *** TABLE ≠.
014600          10  LNE1-NAME   PIC 999.99.
014700          10  LNE1-FL05   PIC X(5).        VALUE ≠ --- ≠.
014800          10  LNE1-FL10   PIC X(15)        VALUE ≠RELATED TO ORIG≠.
014900          10  LNE1-FL15   PIC X(11)        VALUE ≠INAL LABEL ≠.
015000          10  LNE1-ORIG   PIC X(74)        VALUE SPACE.
015100          10  LNE1-FL20   PIC X            VALUE ≠I≠.
015200
015300      05  PRNT-LNE3.
015400          10  LNE3-FL00   PIC XX           VALUE ≠I ≠.
015500          10  LNE3-INFO   PIC X(77)        VALUE SPACE.
015600          10  LNE3-FL01   PIC XXX          VALUE ≠ I ≠.
015700          10  LNE3-FL05.
015800              15  LNE3-RULE PIC X OCCURS 40 TIMES.
015900          10  LNE3-FL10   PIC XX           VALUE ≠ I≠.
016000
016100      05  PRNT-LNE4.
016200          10  LNE4-TABL   PIC 999.99.
016300          10  LNE4-FL01   PIC XX           VALUE SPACE.
016400          10  LNE4-FL05   PIC X(14)        VALUE SPACE.
016500          10  LNE4-FL07   PIC X(112)       VALUE SPACE.
016600          10  LNE4-FL10 REDEFINES LNE4-FL07 OCCURS 14 TIMES.
016700              15  LNE4-RFNX.
016800                  20  LNE4-RFNC  PIC 999.99.
```

```
016900              15  LNE4-FL15   PIC XX.
017000
017100      05  PRNT-LNE5.
017200          10  LNE5-FL01   PIC X          VALUE ≠I≠.
017300          10  LNE5-FL05   PIC X(80)      VALUE ALL ≠-≠.
017400          10  LNE5-FL10   PIC X(42)      VALUE ALL ≠-≠.
017500          10  LNE5-FL15   PIC X          VALUE ≠I≠.
017600
017700      05  PRNT-LNE6.
017800          10  LNE6-FL01   PIC X(81)      VALUE ALL ≠=≠.
017900          10  LNE6-FL05   PIC X(43)      VALUE ALL ≠=≠.
018000
018100      05  PRNT-LNE7.
018200          10  LNE7-FL05   PIC X(14)      VALUE ≠I CONDITIONS =≠.
018300          10  LNE7-CNDS   PIC ZZ9.
018400          10  LNE7-FL10   PIC X(12)      VALUE ≠   ACTIONS =≠.
018500          10  LNE7-ACTS   PIC ZZ9.
018600          10  LNE7-FL15   PIC X(10)      VALUE ≠   RULES =≠.
018700          10  LNE7-RULS   PIC ZZ9.
018800          10  LNE7-FL20   PIC X(78)      VALUE SPACE.
018900          10  LNE7-FL25   PIC X          VALUE ≠I≠.
019000
```

```
019100 PROCEDURE DIVISION USING COMN-INFO.
019200 000-PROGRAM-START SECTION 75.
019300 000-START.
019400
019500     MOVE 3 TO EROR-CODE.
019600     PERFORM 600-ERROR-ROUTINE THRU 699-ERROR-ROUTINE-END.
019700     OPEN INPUT  DTINDEX.
019800     OPEN INPUT DETABLS.
019900     MOVE ≠FIRST≠ TO INDX-KEY.
020000     PERFORM 700-READ-POINTER THRU 799-READ-POINTER-END.
020100     MOVE 0 TO TABL-KEY.
020200     MOVE SPACE TO F-FLAG.
020300     MOVE SPACE TO H-FLAG.
020400     MOVE +0 TO INDX-GP01.
020500     MOVE 0 TO TOTL-LNES.
020600     MOVE +75 TO LINE-CNTR.
020700     MOVE 0 TO PAGE-CNTR.
020800     MOVE ≠F≠ TO PRNT-CODE.
020900
021000 100-GET-REPT-CODES.
021100
021200     ADD +1 TO INDX-GP01.
021300     IF INDX-GP01 IS GREATER THAN 10
021400         MOVE +0 TO INDX-GP01
021500         GO TO 300-READ-INDEX.
021600     IF ROST-CODE (INDX-GP01) = ≠F≠
021700         MOVE ≠X≠ TO F-FLAG.
021800     IF ROST-CODE (INDX-GP01) = ≠H≠
021900         MOVE ≠X≠ TO H-FLAG.
022000     GO TO 100-GET-REPT-CODES.
022100
022200 300-READ-INDEX.
022300
022400     PERFORM 700-READ-POINTER THRU 799-READ-POINTER-END.
022500     MOVE TABL-ACTS TO LNE7-ACTS.
022600     MOVE TABL-CNDS TO LNE7-CNDS.
022700     MOVE TABL-RULS TO LNE7-RULS.
022800     MOVE TABL-INDX TO TABL-KEY.
022900     PERFORM 800-READ-TABLE THRU 899-READ-TABLE-END.
023000     IF EROR-CODE NOT = 0
023100         MOVE 0 TO EROR-CODE
023200         MOVE 0 TO NAME-NMBR
023300         MOVE SPACE TO NAME-ORIG
023400         GO TO 305-TITLE.
023500     MOVE NTRY-DATA TO NAME-RECD.
023600
023700 305-TITLE.
023800
023900     MOVE NAME-NMBR TO LNE1-NAME.
024000     MOVE NAME-ORIG TO LNE1-ORIG.
024100     IF TABL-CNDS = 0
024200         ADD 8 TO TOTL-LNES
024300     ELSE
024400         ADD 9 TO TOTL-LNES.
024500     ADD TOTL-LNES TO LINE-CNTR.
024600     IF LINE-CNTR IS GREATER THAN 60
```

```
247      024700          PERFORM 500-PAGE-HEAD THRU 599-EXIT.
248      024800      MOVE ≠0≠ TO PRNT-CCTL.
249      024900      MOVE PRNT-LNE6 TO PRNT-DATA.
250      025000      PERFORM 590-WRITE THRU 599-EXIT.
251      025100      MOVE PRNT-LNE1 TO PRNT-DATA.
252      025200      PERFORM 590-WRITE THRU 599-EXIT.
253      025300      MOVE PRNT-LNE7 TO PRNT-DATA.
254      025400      PERFORM 590-WRITE THRU 599-EXIT.
255      025500      IF TABL-KEY = 0
256      025600          GO TO 320-CONDS.
257      025700
258      025800 310-READ-CONTINUES.
259      025900
260      026000      PERFORM 800-READ-TABLE THRU 899-READ-TABLE-END.
261      026100      IF EROR-CODE NOT = 0
262      026200          MOVE 0 TO EROR-CODE
263      026300          GO TO 320-CONDS.
264      026400      MOVE NTRY-DATA TO PRNT-DATA.
265      026500      MOVE ≠I≠ TO PRNT-POS1 PRNT-POSL.
266      026600      PERFORM 590-WRITE THRU 599-EXIT.
267      026700      IF TABL-KEY NOT = 0
268      026800          GO TO 310-READ-CONTINUES.
269      026900
270      027000 320-CONDS.
271      027100
272      027200      MOVE ≠I≠ TO PRNT-POS1 PRNT-POSL.
273      027300      PERFORM 590-WRITE THRU 599-EXIT.
274      027400      IF TABL-COND = 0
275      027500          GO TO 333-ACTION-TABLE.
276      027600      MOVE TABL-COND TO TABL-KEY.
277      027700      MOVE PRNT-LNE5 TO PRNT-DATA.
278      027800      PERFORM 590-WRITE THRU 599-EXIT.
279      027900
280      028000 325-READ-CONDITIONS.
281      028100
282      028200      PERFORM 800-READ-TABLE THRU 899-READ-TABLE-END.
283      028300      IF EROR-CODE NOT = 0
284      028400          MOVE 0 TO EROR-CODE
285      028500          GO TO 330-END-CONDS.
286      028600      MOVE NTRY-DATA TO COND-RECD.
287      028700      MOVE COND-INFO TO LNE3-INFO.
288      028800      MOVE COND-FL01 TO LNE3-FL05.
289      028900      MOVE PRNT-LNE3 TO PRNT-DATA.
290      029000      PERFORM 590-WRITE THRU 599-EXIT.
291      029100      IF TABL-KEY NOT = 0
292      029200          GO TO 325-READ-CONDITIONS.
293      029300
294      029400 330-END-CONDS.
295      029500
296      029600      MOVE ALL ≠-≠ TO LNE3-INFO.
297      029700      MOVE ALL ≠-≠ TO LNE3-FL05.
298      029800      MOVE PRNT-LNE3 TO PRNT-DATA.
299      029900      PERFORM 590-WRITE THRU 599-EXIT.
300      030000      GO TO 335-ACTIONS.
301      030100
302      030200 333-ACTION-TABLE.
```

```
0303     030300
0304     030400      MOVE PRNT-LNE5 TO PRNT-DATA.
0305     030500      PERFORM 590-WRITE THRU 599-EXIT.
0306     030600
0307     030700 335-ACTIONS.
0308     030800
0309     030900      MOVE TABL-ACTN TO TABL-KEY.
0310     031000
0311     031100 340-READ-ACTIONS.
0312     031200
0313     031300      PERFORM 800-READ-TABLE THRU 899-READ-TABLE-END.
0314     031400      IF EROR-CODE NOT = 0
0315     031500          MOVE 0 TO EROR-CODE
0316     031600          GO TO 345-END-ACTIONS.
0317     031700
0318     031800      MOVE NTRY-DATA TO ACTN-RECD.
0319     031900      IF GOTO-CNST = ≠GO TO ≠
0320     032000          MOVE GOTO-NAME TO FXGO-NAME
0321     032100          MOVE FXGO-LINE TO ACTN-INFO.
0322     032200      MOVE ACTN-INFO TO LNE3-INFO.
0323     032300      MOVE ACTN-FL01 TO LNE3-FL05.
0324     032400      MOVE PRNT-LNE3 TO PRNT-DATA.
0325     032500      PERFORM 590-WRITE THRU 599-EXIT.
0326     032600      IF TABL-KEY NOT = 0
0327     032700          GO TO 340-READ-ACTIONS.
0328     032800
0329     032900 345-END-ACTIONS.
0330     033000
0331     033100      MOVE PRNT-LNE6 TO PRNT-DATA.
0332     033200      PERFORM 590-WRITE THRU 599-EXIT.
0333     033300
0334     033400 350-REFERENCES.
0335     033500
0336     033600      IF INDX-KEY NOT = ≠AVAIL≠
0337     033700          GO TO 300-READ-INDEX.
0338     033800      MOVE 0 TO PAGE-CNTR.
0339     033900      MOVE 75 TO LINE-CNTR.
0340     034000      MOVE ≠H≠ TO PRNT-CODE.
0341     034100      MOVE ≠FIRST≠ TO INDX-KEY.
0342     034200      PERFORM 700-READ-POINTER THRU 799-READ-POINTER-END.
0343     034300
0344     034400 355-READ-RFTO.
0345     034500
0346     034600      MOVE 0 TO INDX-GP02.
0347     034700      MOVE ≠REFERS TO ≠ TO LNE4-FL05.
0348     034800      PERFORM 700-READ-POINTER THRU 799-READ-POINTER-END.
0349     034900      MOVE TABL-NAME TO LNE4-TABL.
0350     035000      MOVE TABL-RFTO TO TABL-KEY.
0351     035100
0352     035200 360-READ.
0353     035300
0354     035400      IF TABL-KEY = 0
0355     035500          MOVE ≠ NO TABLES ******≠ TO LNE4-FL07
0356     035600          GO TO 367-PRINTIT.
0357     035700      PERFORM 800-READ-TABLE THRU 899-READ-TABLE-END.
0358     035800      IF EROR-CODE NOT = 0
```

```
359      035900          MOVE 0 TO EROR-CODE RFTO-NUMB
360      036000          MOVE SPACE TO RFTO-INFO
361      036100          GO TO 365-PRINT.
362      036200      MOVE NTRY-DATA TO RFTO-RECD.
363      036300
364      036400  365-PRINT.
365      036500
366      036600      MOVE 0 TO INDX-GP01.
367      036700      PERFORM 400-WRITE-RFTO THRU 449-WRITE-RFTO-END RFTO-NUMB
368      036800          TIMES.
369      036900      IF TABL-KEY NOT = 0
370      037000          GO TO 360-READ.
371      037100
372      037200  367-PRINTIT.
373      037300
374      037400      IF RFTO-NUMB = 0
375      037500          MOVE ≠ NO TABLES *****≠ TO LNE4-FL07.
376      037600      MOVE ≠0≠ TO PRNT-CCTL.
377      037700      PERFORM 510-WRITE THRU 599-EXIT.
378      037800      MOVE TABL-RFBY TO TABL-KEY.
379      037900      MOVE ≠REFERENCED BY≠ TO LNE4-FL05.
380      038000      MOVE 0 TO INDX-GP02.
381      038100
382      038200  370-READ.
383      038300
384      038400      IF TABL-KEY = 0
385      038500          MOVE ≠ NO TABLES *****≠ TO LNE4-FL07
386      038600          GO TO 377-PRINTIT.
387      038700      MOVE 0 TO INDX-GP01.
388      038800      PERFORM 800-READ-TABLE THRU 899-READ-TABLE-END.
389      038900      IF EROR-CODE NOT = 0
390      039000          MOVE 0 TO EROR-CODE RFBY-NUMB
391      039100          MOVE SPACE TO RFBY-INFO
392      039200          GO TO 375-PRINT.
393      039300      MOVE NTRY-DATA TO RFBY-RECD.
394      039400
395      039500  375-PRINT.
396      039600
397      039700      PERFORM 450-WRITE-RFBY THRU 499-WRITE-RFBY-END RFBY-NUMB
398      039800          TIMES.
399      039900      IF TABL-KEY NOT = 0
400      040000          GO TO 370-READ.
401      040100
402      040200  377-PRINTIT.
403      040300
404      040400      IF RFBY-NUMB = 0
405      040500          MOVE ≠ NO TABLES *****≠ TO LNE4-FL07.
406      040600      PERFORM 510-WRITE THRU 599-EXIT.
407      040700      IF INDX-KEY NOT = ≠AVAIL≠
408      040800          GO TO 355-READ-RFTO.
409      040900      MOVE +0 TO RETN-CODE.
410      041000      GO TO 900-TERMINATION.
411      041100
412      041200  400-WRITE-RFTO.
413      041300
414      041400      ADD 1 TO INDX-GP01.
```

```
041500     IF INDX-GP01 IS GREATER THAN 23
041600         MOVE 0 TO INDX-GP01
041700         GO TO 449-WRITE-RFTO-END.
041800     ADD 1 TO INDX-GP02.
041900     IF INDX-GP02 IS GREATER THAN 14
042000         MOVE 1 TO INDX-GP02
042100         PERFORM 510-WRITE THRU 599-EXIT.
042200     IF RFTO-TABX (INDX-GP01) NOT NUMERIC
042300         MOVE RFTO-TABX (INDX-GP01) TO LNE4-RFNX (INDX-GP02)
042400     ELSE
042500         MOVE RFTO-TABL (INDX-GP01) TO LNE4-RFNC (INDX-GP02).
042600
042700 449-WRITE-RFTO-END.
042800
042900     EXIT.
043000
043100 450-WRITE-RFBY.
043200
043300     ADD 1 TO INDX-GP01.
043400     IF INDX-GP01 IS GREATER THAN 23
043500         MOVE 0 TO INDX-GP01
043600         GO TO 499-WRITE-RFBY-END.
043700     ADD 1 TO INDX-GP02.
043800     IF INDX-GP02 IS GREATER THAN 14
043900         MOVE 1 TO INDX-GP02
044000         PERFORM 510-WRITE THRU 599-EXIT.
044100     IF RFBY-TABX (INDX-GP01) NOT NUMERIC
044200         MOVE RFBY-TABX (INDX-GP01) TO LNE4-RFNX (INDX-GP02)
044300     ELSE
044400         MOVE RFBY-TABL (INDX-GP01) TO LNE4-RFNC (INDX-GP02).
044500
044600 499-WRITE-RFBY-END.
044700
044800     EXIT.
044900
045000 500-PAGE-HEAD.
045100
045200     IF F-FLAG = SPACE
045300         GO TO 599-EXIT.
045400     MOVE TOTL-LNES TO LINE-CNTR.
045500     MOVE "1" TO PRNT-CCTL.
045600     MOVE SPACE TO PRNT-DATA.
045700     GO TO 590-WRITE.
045800
045900 510-WRITE.
046000
046100     IF H-FLAG = SPACE
046200         GO TO 599-EXIT.
046300     IF PRNT-CCTL = "0"
046400         ADD 3 TO LINE-CNTR.
046500     IF LINE-CNTR IS GREATER THAN 60
046600         MOVE 4 TO LINE-CNTR
046700         MOVE "1" TO PRNT-CCTL
046800         MOVE SPACE TO PRNT-DATA
046900         MOVE PRNT-RECD TO PASD-INFO
047000         CALL "DETBL91" USING COMN-INFO
```

```
047100          MOVE ≠0≠ TO PRNT-CCTL.
047200      MOVE PRNT-LNE4 TO PRNT-DATA.
047300      MOVE SPACE TO PRNT-LNE4.
047400      GO TO 590-WRITE.
047500
047600 590-WRITE.
047700
047800      IF PRNT-CODE = ≠F≠ AND F-FLAG = SPACE
047900          GO TO 599-EXIT.
048000      IF PRNT-CODE = ≠H≠ AND H-FLAG = SPACE
048100          GO TO 599-EXIT.
048200      MOVE PRNT-RECD TO PASD-INFO.
048300      CALL ≠DETBL91≠ USING COMN-INFO.
048400      MOVE SPACE TO PRNT-CCTL.
048500      MOVE SPACE TO PRNT-DATA.
048600
048700 599-EXIT.
048800
048900      EXIT.
049000
049100 600-ERROR-ROUTINE.
049200
049300      MOVE EROR-MESG (EROR-CODE) TO EROR-TEXT.
049400      MOVE EROR-CODE TO MESG-NMBR.
049500      DISPLAY EROR-AREA.
049600      MOVE SPACE TO EROR-TEXT EROR-VALU.
049700      IF EROR-CODE = 1
049800          MOVE +99 TO RETN-CODE
049900          GO TO 900-TERMINATION.
050000
050100 699-ERROR-ROUTINE-END.
050200
050300      EXIT.
050400
050500 700-READ-POINTER.
050600
050700      READ DTINDEX INVALID KEY
050800          MOVE 1 TO EROR-CODE
050900          MOVE INDX-KEY TO EROR-VALU
051000          PERFORM 600-ERROR-ROUTINE THRU 699-ERROR-ROUTINE-END
051100          GO TO 799-READ-POINTER-END.
051200      MOVE TABL-NEXT TO INDX-KEY.
051300      MOVE 0 TO EROR-CODE.
051400      MOVE TABL-ITMS TO TOTL-LNES.
051500
051600 799-READ-POINTER-END.
051700
051800      EXIT.
051900
052000 800-READ-TABLE.
052100
052200      READ DETABLS INVALID KEY
052300          MOVE 2 TO EROR-CODE
052400          MOVE TABL-KEY TO EROR-VALU
052500          MOVE 0 TO TABL-KEY
052600          PERFORM 600-ERROR-ROUTINE THRU 699-ERROR-ROUTINE-END
```

```
7527     052700          GO TO 899-READ-TABLE-END.
6528     052800      MOVE NTRY-NEXT TO TABL-KEY.
6529     052900      MOVE 0 TO EROR-CODE.
2530     053000
 531     053100 899-READ-TABLE-END.
0532     053200
0533     053300      EXIT.
 534     053400
0535     053500 900-TERMINATION.
 536     053600
 537     053700      CLOSE DTINDEX.
 538     053800      CLOSE DETABLS.
 539     053900
2540     054000      RETURN.
 541     054100
0 LENGTH IS  002463
0R SCM USED
```

```
0001    000100 IDENTIFICATION DIVISION.
0002    000200 PROGRAM-ID. DETBL40.
0003    000300 INSTALLATION. SIR GEORGE WILLIAMS UNIVERSITY.
0004    000400 DATE-COMPILED. 1972.    75/07/18..
0005    000500 ENVIRONMENT DIVISION.
0006    000600 CONFIGURATION SECTION.
0007    000700 SOURCE-COMPUTER. 6400.
0008    000800 OBJECT-COMPUTER. 6400.
0009    000900 INPUT-OUTPUT SECTION.
0010    001000 FILE-CONTROL.
0011    001100     SELECT SOURCE-DECK ASSIGN TO SOURCIN.
0012    001200     SELECT INTER1 ASSIGN TO INTER1.
0013    001300     SELECT INTER2 ASSIGN TO INTER2.
0014    001400
0015    001500 DATA DIVISION.
0016    001600 FILE SECTION.
0017    001700 FD  SOURCE-DECK
0018    001800     BLOCK CONTAINS 1 RECORDS
0019    001900     RECORD CONTAINS 80 CHARACTERS
0020    002000     DATA RECORD IS CARD
0021    002100     LABEL RECORDS ARE OMITTED.
0022    002200
0023    002300 01  CARD.
0024    002400     02  CARD-LABL.
0025    002500         03  FIRST-CDIGIT        PIC X.
0026    002600         88  COMNT           VALUES ≠C≠, ≠*≠, ≠$≠, ≠.≠, ≠F≠.
0027    002700         03  LABEL-CDIGIT        PIC X OCCURS 4 TIMES.
0028    002800     02  FILLER              PIC X.
0029    002900         88  NOT-CONTINUATION    VALUES ≠0≠, SPACES.
0030    003000     02  CARD-BODY           PIC X(66).
0031    003100         88  NULL-CARD       VALUE SPACES.
0032    003200     02 CARD-CHARACTER REDEFINES CARD-BODY OCCURS 66 TIMES PIC X.
0033    003300     02  CARDNO              PIC X(8).
0034    003400
0035    003500 FD  INTER1
0036    003600     LABEL RECORDS ARE OMITTED
0037    003700     BLOCK CONTAINS 20 RECORDS
0038    003800     RECORD CONTAINS 76 CHARACTERS
0039    003900     DATA RECORDS ARE INTCARD, LABEL-RECORD.
0040    004000
0041    004100 01  INTCARD.
0042    004200     02  CHAR1               PIC X.
0043    004300     02  INSTMNT          PIC X(71).
0044    004400     02  INT-CHARACTER REDEFINES INSTMNT OCCURS 71 TIMES PIC X.
0045    004500     02  INT-LENGTH              PIC 9(4).
0046    004600
0047    004700 01  LABEL-RECORD.
0048    004800     02  FILLER          PIC X.
0049    004900     02  EXTRA-PART      PIC 99999.
0050    005000     02  FIRST-PART      PIC X(5).
0051    005100     02  FILLER          PIC X(61).
0052    005200     02  INT-LNTHX       PIC X(4).
0053    005300
0054    005400 FD  INTER2
0055    005500     LABEL RECORDS ARE OMITTED
0056    005600     BLOCK CONTAINS 20 RECORDS
```

```
005700     RECORD CONTAINS 80 CHARACTERS
005800     DATA RECORD IS INTREC2.
005900
006000 01  INTREC2                    PIC X(80).
006100
006200 COMMON-STORAGE SECTION.
006300
006400 01  COMN-INFO.
006500     05  RETN-CODE          PIC S9(4) COMP.
006600     05  PASD-INFO.
006700         10  REPT-CODE      PIC X.
006800         10  REPT-INFO.
006900             15  NONX-LABL.  PIC X(5).
007000             15  NONX-INFO   PIC X(132).
007100     05  REPT-LIST REDEFINES PASD-INFO.
007200         10  RQST-CODE      PIC X              OCCURS 10 TIMES.
007300         10  REPT-FLDS      PIC X(128).
007400
007500 WORKING-STORAGE SECTION.
007600 77  LABL-NUMBR                      PIC 9(5) USAGE COMP-2 VALUE 0.
007700 77  CNTR-GP01                  PIC 99   USAGE COMP-2 VALUE 0.
007800 77  L                              PIC 99  USAGE COMP-2.
007900 77  L1                             PIC 99  USAGE COMP-2.
008000 77  J                              PIC 99  USAGE COMP-2.
008100 77  J1                         PIC 99          VALUE 0.
008200 77  I                              PIC 99  USAGE COMP-2.
008300 77  I1                                 PIC 99  USAGE COMP-2.
008400 77  I2                                 PIC 99  USAGE COMP-2.
008500 77  K                                  PIC 99  USAGE COMP-2.
008600 77  SORC-FLAG                          PIC X VALUE ≠0≠.
008700 77  LABL-FLAG                          PIC X  VALUE ≠0≠.
008800 77  GOTO-FLAG                  PIC X           VALUE ≠1≠.
008900 77  DLRS-FLAG                  PIC X           VALUE ≠0≠.
009000 77  LAST-CHAR                  PIC X           VALUE ≠ ≠.
009100 77  EROR-INDX                  PIC S99         VALUE +0 COMP.
009200 77  INDX-GP01                  PIC S9(4)       VALUE +0 COMP.
009300 77  INDX-GP02                  PIC S9(4)       VALUE +0 COMP.
009400 77  C-FLAG                     PIC X           VALUE SPACE.
009500 77  E-FLAG                     PIC X           VALUE SPACE.
009600 77  SORC-LNTH-STOR             PIC 9(4)        COMP VALUE 0.
009700 77  PROG-ENDF                  PIC 9           VALUE 0.
009800
009900 01  LABEL-TABLE.
010000     02  TABLE-LENGTH               PIC 999 USAGE COMP   VALUE 1.
010100     02  LABELS                     OCCURS  500 TIMES.
010200         03  OLDLABL                PIC X(5).
010300         03  ADITION            PIC S99999 USAGE COMP.
010400
010500 01  PACK-TABLE.
010600     02  PAKDCARD                   PIC X   OCCURS 8 TIMES.
010700
010800 01  PAKDR                          REDEFINES PACK-TABLE.
010900     02  PAKD7.
011000         03  PAKD6.
011100             04  PAKD5.
011200                 05  PAKD4.
```

```
011300                          06  PAKD3.
011400                              07  PAKD2.
011500                                  08  PAKD1    PIC X.
011600                                  08  FILLER   PIC X.
011700                              07  FILLER       PIC X.
011800                          06  FILLER           PIC X.
011900                      05  FILLER               PIC X.
012000                  04  FILLER                   PIC X.
012100              03  FILLER                       PIC X.
012200          02  FILLER                           PIC X.
012300
012400 01  LAST-LABEL.
012500      02  FILLER              PIC X    VALUE ≠N≠.
012600      02  LAST-ADITION        PIC 99999    VALUE 0.
012700      02  LAST-OLDLABL        PIC X(5)     VALUE ≠**/**≠.
012800
012900 01  GEN-GOTO.
013000      03  GOTO-CHAR           PIC X            VALUE ≠A≠.
013100      03  LADJGOTO.
013200          05  FILLER                  PIC X(6)     VALUE ≠GO TO ≠.
013300          05  GOTO-ADR.
013400              07  GOTO-NEW            PIC 99999.
013500              07  GOTO-OLD            PIC X(5).
013600          05  GOTO-DIGT REDEFINES GOTO-ADR PIC X OCCURS 10 TIMES.
013700
013800 01  COND-REC.
013900      02  FILLER              PIC X      VALUE ≠C≠.
014000      02  CONDTION            PIC X(71).
014100      02  COND-CHAR REDEFINES CONDTION PIC X OCCURS 71 TIMES.
014200      02  COND-LENGTH         PIC 9(4)   VALUE 71.
014300
014400 01  RESULT-REC.
014500      02  FILLER              PIC X.     VALUE ≠R≠.
014600      02  RESALT              PIC X(71).
014700      02  RES-CHAR REDEFINES RESALT PIC X OCCURS 71 TIMES.
014800      02  RES-LENGTH          PIC 9(4)   VALUE 71.
014900      02  RES-LNTHX REDEFINES RES-LENGTH PIC X(4).
015000
015100 01  ELSE-REC.
015200      02  FILLER              PIC X.     VALUE ≠E≠.
015300      02  ELSE-FIELD          PIC X(71).
015400      02  ELSE-LENGTH         PIC 9(4)   VALUE 71.
015500      02  ELSE-LNTHX REDEFINES ELSE-LENGTH PIC X(4).
015600
015700 01  COMPUTED-GO-TO.
015800      02  GO-TO-COND OCCURS 50 TIMES PIC x.
015900
016000 01  NUMRIC-COMP-GOTO REDEFINES COMPUTED-GO-TO.
016100      02  NUM-GO-TO-COND OCCURS 50 TIMES PIC 9.
016200
016300 01  GOTO-NEXT.
016400      03  FILLER                      PIC X(6)  VALUE ≠GO TO ≠.
016500      03  GOTO-NEWLABL        PIC 99999.
016600      03  GOTO-OLDLABL        PIC X(5).
016700      03  GOTO-OLDDIG  REDEFINES GOTO-OLDLABL OCCURS 5 TIMES PIC
016800
```

```
0169     016900 01  CONDS.
0170     017000     02  FILLER2          PIC X(5)        VALUE ≠.EQ.0≠.
0171     017100     02  EQUAL REDEFINES FILLER2 PIC X OCCURS 5 TIMES.
0172     017200     02  FILLER3          PIC X(5)        VALUE ≠.LT.0≠.
0173     017300     02  LESTHAN REDEFINES FILLER3 PIC X OCCURS 5 TIMES.
0174     017400
0175     017500 01  EXTRA-RECORD.
0176     017600     02  FILLER           PIC X     VALUE ≠A≠.
0177     017700     02  FILLER4          PIC X(71).
0178     017800     02  EXTRA-CHARACTER REDEFINES FILLER4 OCCURS 71 TIMES PIC X.
0179     017900     02  EXTRA-LENGTH             PIC 9(4).
0180     018000
0181     018100 01  CMNT-TABL.
0182     018200     05  CMNT-LNTH       PIC 9(4)        VALUE 0 COMP.
0183     018300     05  CMNT-RECD       PIC X(72) OCCURS 20 TIMES.
0184     018400
0185     018500 01  DO-STACK.
0186     018600     02  STACK-LENGTH             PIC 99  VALUE 0 COMP.
0187     018700     02  DO-TEST OCCURS 99 TIMES.
0188     018800         03  DO-TST               PIC X(67).
0189     018900         03  DO-COND REDEFINES DO-TST OCCURS 67 TIMES PIC X.
0190     019000         03  DO-COND-LENGTH       PIC 99.
0191     019100         03  DO-RSLT              PIC X(67).
0192     019200         03  DO-INCR REDEFINES DO-RSLT OCCURS 67 TIMES PIC X.
0193     019300         03  DO-INCR-LENGTH       PIC 99.
0194     019400         03  DO-PTTN.
0195     019500             05  DO-ADITION  PIC 99999.
0196     019600             05  DO-OLDLABL  PIC X(5).
0197     019700         03  DO-END-LABL          PIC 9(5).
0198     019800     03  DO1 REDEFINES DO-END-LABL.
0199     019900         05  FILLER               PIC 9(4).
0200     020000         05  DO-END               PIC 9.
0201     020100
0202     020200 01  SOURCE-STATEMENT.
0203     020300     03  LABL.
0204     020400         05  FIRST-DIGIT          PIC X.
0205     020500         05  LABEL-DIGIT OCCURS 4 TIMES      PIC X.
0206     020600     03  DIJIT REDEFINES LABL PIC X OCCURS 5 TIMES.
0207     020700     03  SOURCE-CRDS              PIC 99  COMP VALUE 0.
0208     020800     03  SOURCE-LENGTH            PIC 9(4) COMP VALUE 0.
0209     020900     03  BODY.
0210     021000         05  SOURCE-CHARACTER OCCURS 1320 TIMES PIC X.
0211     021100         05  FILLER               PIC X.
0212     021200     03  ALT-BODY REDEFINES BODY.
0213     021300         05  SOURCE-CRD-STORE OCCURS 20 TIMES PIC X(66).
0214     021400         05  FILLER               PIC X.
0215     021500
0216     021600 01  MESG-LINE.
0217     021700     05  MESG-IDNT       PIC X(9)        VALUE ≠*** DET40≠.
0218     021800     05  MESG-NMBR       PIC 99          VALUE 0.
0219     021900     05  MESG-FLDS       PIC X(5)        VALUE ≠ --- ≠.
0220     022000     05  MESG-AREA       PIC X(30)       VALUE SPACE.
0221     022100     05  ERDR-VALU       PIC X(87)       VALUE SPACE.
0222     022200
0223     022300 01  ERDR-MSGS.
0224     022400     05  MESG-VALU.
```

```
225     022500          10  MSG01 PIC X(30) VALUE ≠SYNTAX ERROR IN COMPUTED GO TO≠
226     022600          10  MSG02 PIC X(30) VALUE ≠STATEMENT SYNTAX ERROR        ≠
227     022700          10  MSG03 PIC X(30) VALUE ≠TOO MANY CONTINUATION CARDS   ≠
228     022800          10  MSG04 PIC X(30) VALUE ≠LABEL NOT FOUND               ≠
229     022900          10  MSG05 PIC X(30) VALUE ≠FORTRAN TRANSLATION REQUESTED ≠
230     023000          10  MSG06 PIC X(30) VALUE ≠COMMENT LOST - CAP. EXCEEDED  ≠
231     023100      05  EROR-MESG REDEFINES MESG-VALU OCCURS 6 TIMES PIC X(30).
232     023200
```

```
233      023300 PROCEDURE DIVISION USING COMN-INFO.
234      023400 000-PROGRAM-START SECTION 60.
235      023500 100-START.
236      023600     MOVE +5 TO EROR-INDX.
237      023700     PERFORM 1040-ERROR-ROUTINE THRU 1049-ERROR-ROUTINE-END.
238      023800     MOVE SPACE TO E-FLAG.
239      023900     MOVE SPACE TO C-FLAG.
240      024000     MOVE +0 TO INDX-GP01.
241      024100     MOVE SPACE TO CMNT-LABL.
242      024200     MOVE 0 TO CMNT-LNTH.
243      024300
244      024400 102-GET-REPT-CODES.
245      024500     ADD +1 TO INDX-GP01.
246      024600     IF INDX-GP01 IS GREATER THAN +10
247      024700         GO TO 104-PROCESS.
248      024800     IF RQST-CODE (INDX-GP01) = ≠C≠
249      024900         MOVE ≠X≠ TO C-FLAG.
250      025000     IF RQST-CODE (INDX-GP01) = ≠E≠
251      025100         MOVE ≠X≠ TO E-FLAG.
252      025200     GO TO 102-GET-REPT-CODES.
253      025300
254      025400 104-PROCESS.
255      025500     MOVE ≠00000≠ TO OLDLABL (1).
256      025600     MOVE 0 TO ADITION (1).
257      025700     OPEN INPUT SOURCE-DECK.
258      025800     OPEN OUTPUT INTER1.
259      025900     OPEN OUTPUT INTER2.
260      026000
261      026100 105-READ.
262      026200     READ SOURCE-DECK, AT END GO TO 295-CLOSE.
263      026300     MOVE CARD TO REPT-INFO.
264      026400     PERFORM 1005-WRITE-LINE THRU 1009-WRITE-LINE-END.
265      026500     IF COMNT
266      026600         PERFORM 1010-COMMENT THRU 1014-EXIT
267      026700         GO TO 105-READ.
268      026800     IF NULL-CARD
269      026900         GO TO 105-READ.
270      027000
271      027100 110-READ.
272      027200     IF DLRS-FLAG = ≠1≠
273      027300         GO TO 124-SHIFT.
274      027400     PERFORM 1050-WRITE-COMMENTS THRU 1059-WRITE-COMMENTS-END.
275      027500     IF PROG-ENDF = ≠1≠
276      027600         GO TO 295-CLOSE.
277      027700     IF SORC-FLAG = ≠1≠ GO TO 295-CLOSE.
278      027800     MOVE 0 TO SOURCE-LENGTH, SOURCE-CRDS.
279      027900     MOVE SPACE TO BODY.
280      028000     PERFORM 953-CHECK-DO-STACK THRU 955-EXIT.
281      028100     MOVE CARD-LABL TO LABL.
282      028200
283      028300 112-READ.
284      028400     PERFORM 957-STORE-SOURCE THRU 959-EXIT.
285      028500
286      028600 113-READ.
287      028700     READ SOURCE-DECK, AT END
288      028800         MOVE ≠1≠ TO SORC-FLAG
```

```
0289    028900          GO TO 114-FIND-LENGTH.
0290    029000      MOVE CARD TO REPT-INFO.
0291    029100      PERFORM 1005-WRITE-LINE THRU 1009-WRITE-LINE-END.
0292    029200      IF COMNT
0293    029300          PERFORM 1010-COMMENT THRU 1014-EXIT
0294    029400          GO TO 113-READ.
0295    029500      IF NULL-CARD
0296    029600          GO TO 113-READ.
0297    029700      IF NOT-CONTINUATION
0298    029800          NEXT SENTENCE
0299    029900      ELSE
0300    030000          GO TO 112-READ.
0301    030100
0302    030200 114-FIND-LENGTH.
0303    030300      IF SOURCE-CHARACTER (SOURCE-LENGTH) = SPACE
0304    030400          SUBTRACT 1 FROM SOURCE-LENGTH
0305    030500          GO TO 114-FIND-LENGTH.
0306    030600
0307    030700 115-PACK-CARD.
0308    030800      MOVE 0 TO I, J.
0309    030900      MOVE SPACES TO PACK-TABLE.
0310    031000
0311    031100 117-PACK-CARD.
0312    031200      ADD 1 TO I.
0313    031300      IF I IS GREATER THAN SOURCE-LENGTH
0314    031400          GO TO 120-IDENTIFY-NONEXEC.
0315    031500      IF SOURCE-CHARACTER (I) = SPACE
0316    031600          GO TO 117-PACK-CARD.
0317    031700      IF J IS GREATER THAN 7 GO TO 120-IDENTIFY-NONEXEC.
0318    031800      ADD 1 TO J.
0319    031900      MOVE SOURCE-CHARACTER (I) TO PAKDCARD (J).
0320    032000      GO TO 117-PACK-CARD.
0321    032100
0322    032200 120-IDENTIFY-NONEXEC.
0323    032300      IF PAKD7 NOT = ≠FORMAT(≠
0324    032400          GO TO 122-DATA-STMNT.
0325    032500      EXAMINE BODY TALLYING UNTIL FIRST ≠=≠.
0326    032600      IF TALLY IS GREATER THAN SOURCE-LENGTH.
0327    032700          GO TO 140-NONEXEC.
0328    032800      MOVE 6 TO I
0329    032900      MOVE 0 TO J.
0330    033000      PERFORM 900-COUNT-BRACKETS UNTIL I = TALLY.
0331    033100      IF J = 0 GO TO 160-VALID-STMNT
0332    033200      ELSE
0333    033300          GO TO 140-NONEXEC.
0334    033400
0335    033500 122-DATA-STMNT.
0336    033600      IF PAKD4 NOT = ≠DATA≠
0337    033700          GO TO 123-CHECK-DOLLARS.
0338    033800      EXAMINE BODY TALLYING UNTIL FIRST ≠=≠.
0339    033900      IF TALLY IS GREATER THAN SOURCE-LENGTH
0340    034000          GO TO 140-NONEXEC.
0341    034100      MOVE 4 TO I.
0342    034200      MOVE 0 TO J.
0343    034300      PERFORM 900-COUNT-BRACKETS UNTIL I = TALLY.
0344    034400      IF J NOT = 0
```

```
345     034500          GO TO 140-NONEXEC.
346     034600      ADD 1 TO I.
347     034700      EXAMINE BODY TALLYING UNTIL FIRST ≠/≠.
348     034800      IF TALLY IS LESS THAN I
349     034900          GO TO 140-NONEXEC.
350     035000      GO TO 160-VALID-STMNT.
351     035100
352     035200 123-CHECK-DOLLARS.
353     035300      EXAMINE BODY TALLYING UNTIL FIRST ≠$≠.
354     035400      IF TALLY IS GREATER THAN SOURCE-LENGTH
355     035500          GO TO 127-MISC-STMTS.
356     035600      MOVE SOURCE-LENGTH TO SORC-LNTH-STOR.
357     035700      MOVE ≠1≠ TO DLRS-FLAG.
358     035800      MOVE TALLY TO SOURCE-LENGTH.
359     035900      GO TO 127-MISC-STMTS.
360     036000
361     036100 124-SHIFT.
362     036200      MOVE SOURCE-LENGTH TO I.
363     036300      MOVE ≠0≠ TO DLRS-FLAG.
364     036400      ADD 1 TO I.
365     036500      MOVE 0 TO J.
366     036600
367     036700 125-MOVE-BACK.
368     036800      ADD 1 TO I.
369     036900      IF I IS GREATER THAN SORC-LNTH-STOR
370     037000          MOVE J TO SOURCE-LENGTH
371     037100          GO TO 126-BLANK-REST.
372     037200      ADD 1 TO J.
373     037300      MOVE SOURCE-CHARACTER (I) TO SOURCE-CHARACTER (J).
374     037400      GO TO 125-MOVE-BACK.
375     037500
376     037600 126-BLANK-REST.
377     037700      ADD 1 TO J.
378     037800      IF J IS GREATER THAN SORC-LNTH-STOR
379     037900          GO TO 115-PACK-CARD.
380     038000      MOVE SPACE TO SOURCE-CHARACTER (J).
381     038100      GO TO 126-BLANK-REST.
382     038200
383     038300 127-MISC-STMTS.
384     038400      IF PAKD8 = ≠NAMELIST≠ OR ≠BLOCKDAT≠ OR ≠EXTERNAL≠ OR
385     038500      ≠DIMENSIO≠ OR ≠EQUIVALE≠ OR ≠TYPELOGI≠ OR ≠TYPEINTE≠ OR
386     038600      ≠TYPEREAL≠ OR ≠TYPECHAR≠ OR ≠TYPECOMP≠ OR ≠TYPEDOUB≠ OR
387     038700      ≠DOUBLEPR≠ OR ≠FORTRANV≠ OR ≠FORTRANI≠ OR ≠FUNCTION≠ OR
388     038800      ≠SUBROUTI≠ OR ≠CHARACTF≠ OR ≠REALFUNC≠ OR ≠INTEGERF≠ OR
389     038900      ≠DOUBLEFU≠ OR ≠COMPLEXF≠ OR ≠LOGICALF≠
390     039000          GO TO 140-NONEXEC.
391     039100      IF PAKD7 = ≠COMMON/≠
392     039200          GO TO 140-NONEXEC.
393     039300      IF PAKD7 = ≠LOGICAL≠ OR ≠INTEGER≠ OR ≠COMPLEX≠ OR ≠PROGRAM≠
394     039400      AND    PAKDCARD (8) IS ALPHABETIC
395     039500          GO TO 140-NONEXEC.
396     039600      IF PAKD4 = ≠REAL≠ GO TO 129-FIND-EQUALS.
397     039700      IF PAKD5 = ≠ENTRY≠ GO TO 129-FIND-EQUALS.
398     039800      IF PAKD6 = ≠DOUBLE≠ OR ≠COMMON≠ GO TO 129-FIND-EQUALS.
399     039900      IF PAKD4 = ≠END ≠
400     040000          MOVE ≠1≠ TO PROG-ENDF.
```

```
0401    040100      GO TO 160-VALID-STMNT.
0402    040200
0403    040300  129-FIND-EQUALS.
0404    040400      EXAMINE BODY TALLYING UNTIL FIRST ≠=≠.
0405    040500      IF TALLY IS GREATER THAN SOURCE-LENGTH
0406    040600          GO TO 140-NONEXEC
0407    040700      ELSE
0408    040800          GO TO 160-VALID-STMNT.
0409    040900
0410    041000  140-NONEXEC.
0411    041100      MOVE 0 TO I.
0412    041200      MOVE LABL TO NONX-LABL.
0413    041300      PERFORM 1015-WRITE-NONEXEC THRU 1019-END.
0414    041400      MOVE 0 TO I.
0415    041500      GO TO 110-READ.
0416    041600
0417    041700  160-VALID-STMNT.
0418    041800      IF LABL = SPACES GO TO 165-NON-LABELLED.
0419    041900      EXAMINE LABL REPLACING LEADING ≠ ≠ BY ≠0≠.
0420    042000      IF LABEL-DIGIT (4) = SPACES PERFORM 905-SQUEEZE-RIGHT THRU


0452    045200  205-STATEMENT.
0453    045300      MOVE ≠A≠ TO CHAR1.
0454    045400      MOVE 0 TO I.
0455    045500      PERFORM 961-WRITE-INTCARD THRU 965-EXIT.
0456    045600      GO TO 110-READ.
```

```
045700
045800 210-CHECK-IF.
045900     EXAMINE BODY TALLYING UNTIL FIRST ≠=≠.
046000     IF TALLY IS GREATER THAN SOURCE-LENGTH GO TO 213-IDENTIFY-IF
046100     IF PAKDCARD (3) NOT = ≠(≠ GO TO 205-STATEMENT.
046200     MOVE 2 TO I.
046300     MOVE 0 TO J.
046400     PERFORM 900-COUNT-BRACKETS UNTIL I = TALLY
046500     IF J NOT = 0 GO TO 218-IDENTIFY-IF.
046600     EXAMINE BODY TALLYING UNTIL FIRST ≠(≠.
046700     MOVE TALLY TO I.
046800     ADD 1 TO I.
046900     MOVE 1 TO J.
047000     PERFORM 900-COUNT-BRACKETS UNTIL J = 0.
047100
047200 211-ONE-BRANCH-IF.
047300     PERFORM 912-INCR-I.
047400     IF SOURCE-CHARACTER (I) = SPACES GO TO 211-ONE-BRANCH-IF.
047500     IF SOURCE-CHARACTER (I) = ≠=≠ GO TO 205-STATEMENT.
047600     IF LABL = SPACES PERFORM 910-GENERATE-LABEL.
047700     MOVE I TO L1.
047800     PERFORM 915-WRITE-CONDITION.
047900     WRITE INTCARD FROM COND-REC.
048000     MOVE L1 TO I.
048100     PERFORM 930-WRITE-RESULT.
048200     PERFORM 977-FIND-NEXT-LABEL THRU 980-EXIT.
048300     MOVE GOTO-NEXT TO ELSE-FIELD.
048400     MOVE 22 TO ELSE-LENGTH.
048500     WRITE INTCARD FROM ELSE-REC.
048600     MOVE ≠0≠ TO LABL-FLAG.
048700     IF CARD-LABL = SPACE
048800         PERFORM 910-GENERATE-LABEL.
048900     GO TO 110-READ.
049000
049100 213-IDENTIFY-IF.
049200     IF PAKDCARD (3) = ≠(≠ GO TO 218-IDENTIFY-IF.
049300     IF PAKD8 NOT = ≠IFACCUMU≠ GO TO 214-IDENTIFY-IF.
049400     MOVE ≠ACCUMULATOR OVERFLOW≠ TO CONDTION.
049500     MOVE 22 TO COND-LENGTH.
049600     MOVE 21 TO I.
049700     GO TO 216-IF-OK.
049800
049900 214-IDENTIFY-IF.
050000     IF PAKD8 NOT = ≠IFQUOTIE≠ GO TO 215-IDENTIFY-IF.
050100     MOVE ≠QUOTIENT OVERFLOW≠ TO CONDTION.
050200     MOVE 19 TO COND-LENGTH.
050300     MOVE 18 TO I.
050400     GO TO 216-IF-OK.
050500
050600 215-IDENTIFY-IF.
050700     IF PAKD8 NOT = ≠IFDIVIDE≠ GO TO 205-STATEMENT.
050800     MOVE ≠DIVIDE CHECK≠ TO CONDTION.
050900     MOVE 14 TO COND-LENGTH.
051000     MOVE 13 TO I.
051100
051200 216-IF-OK.
```

```
513      051300      MOVE 1 TO J.
514      051400      MOVE SPACES TO GOTO-ADR.
515      051500      PERFORM 940-EXTRACT-ADR THRU 941-EXIT
516      051600      UNTIL SOURCE-CHARACTER (I) = ≠,≠.
517      051700      MOVE LADJGOTO TO RESALT.
518      051800      MOVE 16 TO RES-LENGTH.
519      051900      MOVE SPACES TO GOTO-ADR.
520      052000      PERFORM 940-EXTRACT-ADR THRU 941-EXIT UNTIL I = SOURCE-LENGT
521      052100      MOVE LADJGOTO TO ELSE-FIELD.
522      052200      MOVE 16 TO ELSE-LENGTH.
523      052300      IF LABL = SPACE PERFORM 910-GENERATE-LABEL.
524      052400      WRITE INTCARD FROM COND-REC.
525      052500      WRITE INTCARD FROM RESULT-REC.
526      052600      WRITE INTCARD FROM ELSE-REC.
527      052700      MOVE ≠0≠ TO LABL-FLAG.
528      052800      GO TO 110-READ.
529      052900
530      053000  218-IDENTIFY-IF.
531      053100      EXAMINE BODY TALLYING UNTIL FIRST ≠(≠.
532      053200      MOVE TALLY TO I.
533      053300      ADD 1 TO I.
534      053400      MOVE 1 TO J.
535      053500      MOVE SPACE TO PACK-TABLE.
536      053600
537      053700  220-PACK-CONDITION.
538      053800      ADD 1 TO I.
539      053900      IF I IS GREATER THAN SOURCE-LENGTH
540      054000          GO TO 223-CONDITION-PACKED.


541      054100      IF SOURCE-CHARACTER (I) = SPACES GO TO 220-PACK-CONDITION.
542      054200      MOVE SOURCE-CHARACTER (I) TO PAKDCARD (J).
543      054300      ADD 1 TO J.
544      054400      IF J IS GREATER THAN 8 GO TO 223-CONDITION-PACKED.
545      054500      GO TO 220-PACK-CONDITION.
546      054600
547      054700  223-CONDITION-PACKED.
548      054800      IF PAKD4 = ≠EOF≠
549      054900          MOVE ≠EOF≠ TO CONDTION
550      055000          MOVE 5 TO COND-LENGTH
551      055100          GO TO 225-IF-OK.
552      055200      IF PAKD5 = ≠UNIT.≠ GO TO 224-UNIT-COND.
553      055300      IF PAKD7 = ≠ENDFILE≠
554      055400          MOVE ≠ENDFILE≠ TO CONDTION
555      055500          MOVE 9 TO COND-LENGTH
556      055600          GO TO 225-IF-OK.
557      055700      IF PAKD8 = ≠SENSELIG≠
558      055800          MOVE ≠SENSE LIGHT≠ TO CONDTION
559      055900          MOVE 13 TO COND-LENGTH
560      056000          GO TO 225-IF-OK.
561      056100      IF PAKD8 = ≠SENSESWI≠
562      056200          MOVE ≠SENSE SWITCH≠ TO CONDTION
563      056300          MOVE 14 TO COND-LENGTH
564      056400          GO TO 225-IF-OK.
565      056500      GO TO 227-IDENTIFY-IF.
566      056600
567      056700  224-UNIT-COND.
568      056800      MOVE ≠UNIT NOT READY≠ TO CONDTION.
```

```
569      056900      MOVE 14 TO COND-LENGTH.
570      057000      EXAMINE BODY TALLYING  UNTIL FIRST ≠)≠.
571      057100      MOVE TALLY TO I.
572      057200      ADD 1 TO I.
573      057300      MOVE SPACES TO GOTO-ADR.
574      057400      MOVE 1 TO J.
575      057500      PERFORM 940-EXTRACT-ADR THRU 941-EXIT UNTIL
576      057600          SOURCE-CHARACTER (I) = ≠.≠.
577      057700      PERFORM 979-NEXT-IS-GENERATED THRU 980-EXIT.
578      057800      MOVE GOTO-NEXT TO ELSE-FIELD.
579      057900      MOVE 16 TO ELSE-LENGTH.
580      058000      MOVE LADJGOTO TO RESALT.
581      058100      MOVE 16 TO RES-LENGTH.
582      058200      IF LABL = SPACE PERFORM 910-GENERATE-LABEL.
583      058300      WRITE INTCARD FROM COND-REC.
584      058400      WRITE INTCARD FROM RESULT-REC.
585      058500      WRITE INTCARD FROM ELSE-REC.
586      058600      MOVE ≠0≠ TO LABL-FLAG.
587      058700      MOVE ≠UNIT READY AND NO PREVIOUS ERROR≠ TO CONDTION.
588      058800      MOVE 32 TO COND-LENGTH.
589      058900      MOVE SPACES TO GOTO-ADR.
590      059000      MOVE 1 TO J.
591      059100      PERFORM 940-EXTRACT-ADR THRU 941-EXIT.
592      059200      PERFORM 940-EXTRACT-ADR THRU 941-EXIT UNTIL
593      059300          SOURCE-CHARACTER (I) = ≠.≠.
594      059400      MOVE LADJGOTO TO RESALT.
595      059500      MOVE 16 TO RES-LENGTH.
596      059600      PERFORM 979-NEXT-IS-GENERATED THRU 980-EXIT.
597      059700      MOVE GOTO-NEXT TO ELSE-FIELD.
598      059800      MOVE 16 TO ELSE-LENGTH.
599      059900      PERFORM 910-GENERATE-LABEL.
600      060000      WRITE INTCARD FROM COND-REC.
601      060100      WRITE INTCARD FROM RESULT-REC.
602      060200      WRITE INTCARD FROM ELSE-REC.
603      060300      MOVE ≠0≠ TO LABL-FLAG.
604      060400      MOVE ≠EOF SENSED ON UNIT≠ TO CONDTION.
605      060500      MOVE 18 TO COND-LENGTH.
606      060600      MOVE SPACES TO GOTO-ADR.
607      060700      MOVE 1 TO J.
608      060800      PERFORM 940-EXTRACT-ADR THRU 941-EXIT.
609      060900      PERFORM 940-EXTRACT-ADR THRU 941-EXIT UNTIL
610      061000          SOURCE-CHARACTER (I) = ≠.≠.
611      061100      MOVE LADJGOTO TO RESALT.
612      061200      MOVE 16 TO RES-LENGTH.
613      061300      PERFORM 979-NEXT-IS-GENERATED THRU 980-EXIT.
614      061400      MOVE GOTO-NEXT TO ELSE-FIELD.
615      061500      MOVE 16 TO ELSE-LENGTH.
616      061600      PERFORM 910-GENERATE-LABEL.
617      061700      WRITE INTCARD FROM COND-REC.
618      061800      WRITE INTCARD FROM RESULT-REC.
619      061900      WRITE INTCARD FROM ELSE-REC.
620      062000      MOVE ≠0≠ TO LABL-FLAG.
621      062100      MOVE ≠PARITY ERROR ON UNIT≠ TO CONDTION.
622      062200      MOVE 16 TO COND-LENGTH.
623      062300      MOVE SPACES TO GOTO-ADR.
624      062400      MOVE 1 TO J.
```

```
0625    062500      PERFORM 940-EXTRACT-ADR THRU 941-EXIT
0626    062600          UNTIL I = SOURCE-LENGTH.
0627    062700      MOVE LADJGOTO TO RESALT.
0628    062800      MOVE 16 TO RES-LENGTH.
0629    062900      PERFORM 977-FIND-NEXT-LABEL THRU 980-EXIT.
0630    063000      MOVE GOTO-NEXT TO ELSE-FIELD.
0631    063100      MOVE 16 TO ELSE-LENGTH.
0632    063200      PERFORM 910-GENERATE-LABEL.
0633    063300      WRITE INTCARD FROM COND-REC.
0634    063400      WRITE INTCARD FROM RESULT-REC.
0635    063500      WRITE INTCARD FROM ELSE-REC.
0636    063600      MOVE ≠0≠ TO LABL-FLAG.
0637    063700      GO TO 110-READ.
0638    063800
0639    063900  225-IF-OK.
0640    064000      EXAMINE BODY TALLYING UNTIL FIRST ≠)≠.
0641    064100      MOVE TALLY TO I.
0642    064200      ADD 1 TO I.
0643    064300      MOVE 1 TO J.
0644    064400      MOVE SPACES TO GOTO-ADR.
0645    064500      PERFORM 940-EXTRACT-ADR THRU 941-EXIT.
0646    064600      PERFORM 940-EXTRACT-ADR THRU 941-EXIT UNTIL
0647    064700          SOURCE-CHARACTER (I) = ≠.≠.
0648    064800      MOVE LADJGOTO TO RESALT.
0649    064900      MOVE 16 TO RES-LENGTH.
0650    065000      MOVE SPACES TO GOTO-ADR.
0651    065100      MOVE 1 TO J.
0652    065200      PERFORM 940-EXTRACT-ADR THRU 941-EXIT UNTIL
0653    065300          I = SOURCE-LENGTH.
0654    065400      MOVE LADJGOTO TO ELSE-FIELD.
0655    065500      MOVE 16 TO ELSE-LENGTH.
0656    065600      IF LABL = SPACE PERFORM 910-GENERATE-LABEL.
0657    065700      WRITE INTCARD FROM COND-REC.
0658    065800      WRITE INTCARD FROM RESULT-REC.
0659    065900      WRITE INTCARD FROM ELSE-REC.
0660    066000      MOVE ≠0≠ TO LABL-FLAG.
0661    066100      GO TO 110-READ.
0662    066200
0663    066300  227-IDENTIFY-IF.
0664    066400      EXAMINE BODY TALLYING UNTIL FIRST ≠(≠.
0665    066500      MOVE TALLY TO I.
0666    066600      ADD 1 TO I.
0667    066700      MOVE 1 TO J.
0668    066800      PERFORM 900-COUNT-BRACKETS UNTIL J = 0.
0669    066900
0670    067000  228-SKIP-SPACES.
0671    067100      PERFORM 912-INCR-I.
0672    067200      IF SOURCE-CHARACTER (I) = SPACES GO TO 228-SKIP-SPACES.
0673    067300      IF SOURCE-CHARACTER (I) NOT NUMERIC SUBTRACT 1 FROM I.
0674    067400          GO TO 211-ONE-BRANCH-IF.
0675    067500      MOVE I TO J.
0676    067600      MOVE 0 TO L.
0677    067700      PERFORM 943-COUNT-COMMAS UNTIL J = SOURCE-LENGTH.
0678    067800      IF L = 1 GO TO 235-2BRANCH-IF.
0679    067900      PERFORM 915-WRITE-CONDITION.
0680    068000      MOVE L TO L1, K.
```

```
0681      068100      MOVE 0 TO L.
0682      068200      PERFORM 945-WRITE-LESSTHAN 5 TIMES.
0683      068300      MOVE K TO COND-LENGTH.
0684      068400      MOVE 1 TO J.
0685      068500      PERFORM 940-EXTRACT-ADR THRU 941-EXIT.
0686      068600      PERFORM 940-EXTRACT-ADR THRU 941-EXIT UNTIL
0687      068700          SOURCE-CHARACTER (I) = ≠.≠.
0688      068800      MOVE LADJGOTO TO RESALT.
0689      068900      ADD 5 TO J GIVING RES-LENGTH.
0690      069000      IF LABL = SPACE
0691      069100          PERFORM 910-GENERATE-LABEL.
0692      069200      PERFORM 979-NEXT-IS-GENERATED THRU 980-EXIT.
0693      069300      MOVE GOTO-NEXT TO ELSE-FIELD.
0694      069400      MOVE 16 TO ELSE-LENGTH.
0695      069500      WRITE INTCARD FROM COND-REC.
0696      069600      WRITE INTCARD FROM RESULT-REC.
0697      069700      WRITE INTCARD FROM ELSE-REC.
0698      069800      MOVE ≠0≠ TO LABL-FLAG.
0699      069900      MOVE L1 TO K.
0700      070000      MOVE 0 TO L.
0701      070100      PERFORM 947-WRITE-EQUAL 5 TIMES.
0702      070200      MOVE 1 TO J.
0703      070300      MOVE SPACE TO GOTO-ADR.
0704      070400      PERFORM 940-EXTRACT-ADR THRU 941-EXIT.
0705      070500      PERFORM 940-EXTRACT-ADR THRU 941-EXIT UNTIL
0706      070600          SOURCE-CHARACTER (I) = ≠.≠.
0707      070700      MOVE LADJGOTO TO RESALT.
0708      070800      ADD 5 TO J GIVING RES-LENGTH.
0709      070900      MOVE 1 TO J.
0710      071000      MOVE SPACE TO GOTO-ADR.
0711      071100      PERFORM 940-EXTRACT-ADR THRU 941-EXIT UNTIL
0712      071200          I = SOURCE-LENGTH.
0713      071300      MOVE LADJGOTO TO ELSE-FIELD.
0714      071400      ADD 5 TO J GIVING ELSE-LENGTH.
0715      071500      PERFORM 910-GENERATE-LABEL.
0716      071600      WRITE INTCARD FROM COND-REC.
0717      071700      WRITE INTCARD FROM RESULT-REC.
0718      071800      WRITE INTCARD FROM ELSE-REC.
0719      071900      MOVE ≠0≠ TO LABL-FLAG.
0720      072000      GO TO 110-READ.
0721      072100
0722      072200  235-2BRANCH-IF.
0723      072300      PERFORM 915-WRITE-CONDITION.
0724      072400      MOVE 1 TO J.
0725      072500      MOVE SPACE TO GOTO-ADR.
0726      072600      PERFORM 940-EXTRACT-ADR THRU 941-EXIT.
0727      072700      PERFORM 940-EXTRACT-ADR THRU 941-EXIT
0728      072800          UNTIL SOURCE-CHARACTER (I) = ≠.≠.
0729      072900      MOVE LADJGOTO TO RESALT.
0730      073000      ADD 5 TO J GIVING RES-LENGTH.
0731      073100      MOVE 1 TO J.
0732      073200      MOVE SPACE TO GOTO-ADR.
0733      073300      PERFORM 940-EXTRACT-ADR THRU 941-EXIT
0734      073400          UNTIL I = SOURCE-LENGTH.
0735      073500      MOVE LADJGOTO TO ELSE-FIELD.
0736      073600      ADD 5 TO J GIVING ELSE-LENGTH.
```

```
0737    073700      IF LABL = SPACE PERFORM 910-GENERATE-LABEL.
0738    073800      WRITE INTCARD FROM COND-REC.
0739    073900      WRITE INTCARD FROM RESULT-REC.
0740    074000      WRITE INTCARD FROM ELSE-REC.
0741    074100      MOVE ≠0≠ TO LABL-FLAG.
0742    074200      GO TO 110-READ.
0743    074300
0744    074400  240-CHECK-DO.
0745    074500      EXAMINE BODY TALLYING UNTIL FIRST ≠=≠.
0746    074600      MOVE TALLY TO I.
0747    074700      ADD 1 TO I.
0748    074800
0749    074900  243-CHECK-DO.
0750    075000      ADD 1 TO I.
0751    075100      IF I IS GREATER THAN SOURCE-LENGTH
0752    075200          GO TO 205-STATEMENT.
0753    075300      IF SOURCE-CHARACTER (I) = SPACE OR SOURCE-CHARACTER (I)
0754    075400      IS NUMERIC
0755    075500          GO TO 243-CHECK-DO.
0756    075600      IF SOURCE-CHARACTER (I) = ≠.≠ GO TO 245-DO-STATEMENT.
0757    075700      GO TO 205-STATEMENT.
0758    075800
0759    075900  245-DO-STATEMENT.
0760    076000      MOVE SPACES TO FILLER4.
0761    076100      ADD 1 TO STACK-LENGTH.
0762    076200      MOVE SPACES TO DO-TEST (STACK-LENGTH).
0763    076300      MOVE 0 TO DO-END-LABL (STACK-LENGTH).
0764    076400      MOVE 3 TO I.
0765    076500
0766    076600  247-DO-STATEMENT.
0767    076700      IF SOURCE-CHARACTER (I) IS NOT NUMERIC
0768    076800          PERFORM 912-INCR-I
0769    076900          GO TO 247-DO-STATEMENT.
0770    077000
0771    077100  248-DO-STATEMENT.
0772    077200      MOVE SOURCE-CHARACTER (I) TO DO-END (STACK-LENGTH).
0773    077300
0774    077400  249-SKIP-SPACE.
0775    077500      PERFORM 912-INCR-I.
0776    077600      IF SOURCE-CHARACTER (I) = SPACES GO TO 249-SKIP-SPACE.
0777    077700      IF SOURCE-CHARACTER (I) IS NUMERIC
0778    077800          MULTIPLY 10 BY DO-END-LABL (STACK-LENGTH)
0779    077900          GO TO 248-DO-STATEMENT.
0780    078000      MOVE 1 TO J.
0781    078100      MOVE 1 TO K.
0782    078200
0783    078300  251-EXTRACT-VARB.
0784    078400      MOVE SOURCE-CHARACTER (I) TO EXTRA-CHARACTER (J).
0785    078500          DO-COND (STACK-LENGTH, K). DO-INCR (STACK-LENGTH, K).
0786    078600      IF SOURCE-CHARACTER (I) = ≠=≠
0787    078700          MOVE K TO DO-INCR-LENGTH (STACK-LENGTH)
0788    078800          GO TO 255-INITIALIZE.
0789    078900
0790    079000  253-SKIP-SPACE.
0791    079100      PERFORM 912-INCR-I.
0792    079200      IF SOURCE-CHARACTER (I) = SPACE GO TO 253-SKIP-SPACE.
```

```
793      079300      ADD 1 TO J, K.
794      079400      GO TO 251-EXTRACT-VARB.
795      079500
796      079600  255-INITIALIZE.
797      079700      PERFORM 912-INCR-I.
798      079800      IF SOURCE-CHARACTER (I) = ≠.≠
799      079900          MOVE J TO EXTRA-LENGTH
800      080000          GO TO 257-EXTRACT-COND.
801      080100      ADD 1 TO J.
802      080200      MOVE SOURCE-CHARACTER (I) TO EXTRA-CHARACTER (J).
803      080300      GO TO 255-INITIALIZE.
804      080400
805      080500  257-EXTRACT-COND.
806      080600      ADD 1 TO I.
807      080700      IF I IS GREATER THAN SOURCE-LENGTH
808      080800          GO TO 263-EXTRACT-INCR.
809      080900      IF SOURCE-CHARACTER (I) = SPACE
810      081000          GO TO 257-EXTRACT-COND.
811      081100      IF SOURCE-CHARACTER (I) = ≠,≠
812      081200          GO TO 259-EXTRACT-INCR.
813      081300      ADD 1 TO K.
814      081400      MOVE SOURCE-CHARACTER (I) TO DO-COND (STACK-LENGTH, K).
815      081500      GO TO 257-EXTRACT-COND.
816      081600
817      081700  259-EXTRACT-INCR.
818      081800      MOVE K TO DO-COND-LENGTH (STACK-LENGTH).
819      081900      MOVE 1 TO J.
820      082000      MOVE DO-INCR-LENGTH (STACK-LENGTH) TO K.
821      082100
822      082200  260-COPY-VARB.
823      082300      ADD 1 TO K.
824      082400      MOVE DO-INCR (STACK-LENGTH, J) TO DO-INCR (STACK-LENGTH, K).
825      082500      ADD 1 TO J.
826      082600      IF DO-INCR (STACK-LENGTH, J) = ≠=≠ GO TO 261-INCR.
827      082700      GO TO 260-COPY-VARB.
828      082800
829      082900  261-INCR.
830      083000      ADD 1 TO K.
831      083100      MOVE ≠+≠ TO DO-INCR (STACK-LENGTH, K).
832      083200
833      083300  262-INCR.
834      083400      IF I IS GREATER THAN SOURCE-LENGTH
835      083500          GO TO 265-DO-FINISHED.
836      083600      IF SOURCE-CHARACTER (I) = SPACE
837      083700          GO TO 262-INCR.
838      083800      ADD 1 TO K.
839      083900      MOVE SOURCE-CHARACTER (I) TO DO-INCR (STACK-LENGTH, K).
840      084000      GO TO 262-INCR.
841      084100
842      084200  263-EXTRACT-INCR.
843      084300      PERFORM 259-EXTRACT-INCR THRU 261-INCR.
844      084400      ADD 1 TO K.
845      084500      MOVE 1 TO DO-INCR (STACK-LENGTH, K).
846      084600
847      084700  265-DO-FINISHED.
848      084800      WRITE INTCARD FROM EXTRA-RECORD.
```

```
849      084900      MOVE ≠0≠ TO LABL-FLAG.
850      085000      MOVE K TO DO-INCR-LENGTH (STACK-LENGTH).
851      085100      IF CARD-LABL = SPACE
852      085200          PERFORM 910-GENERATE-LABEL.
853      085300      MOVE LAST-OLDLABL TO DO-OLDLABL (STACK-LENGTH).
854      085400      MOVE LAST-ADITION TO DO-ADITION (STACK-LENGTH).
855      085500      GO TO 110-READ.
856      085600
857      085700  270-CHECK-GO.
858      085800      EXAMINE BODY TALLYING UNTIL FIRST ≠=≠.
859      085900      IF TALLY IS LESS THAN SOURCE-LENGTH
860      086000          GO TO 205-STATEMENT.
861      086100      IF PAKDCARD (5) = ≠(≠
862      086200          GO TO 278-COMPUTED-GO-TO.
863      086300      EXAMINE BODY TALLYING UNTIL FIRST ≠(≠
864      086400      IF TALLY IS LESS THAN SOURCE-LENGTH
865      086500          GO TO 271-ASSIGNED-GO-TO.
866      086600      GO TO 205-STATEMENT.
867      086700
868      086800  271-ASSIGNED-GO-TO.
869      086900      MOVE TALLY TO K.
870      087000      MOVE 0 TO J.
871      087100      MOVE SPACE TO PACK-TABLE.
872      087200      MOVE SPACE TO COMPUTED-GO-TO.
873      087300      MOVE 0 TO I.
874      087400
875      087500  272-PACK.
876      087600      ADD 1 TO I.
877      087700      IF SOURCE-CHARACTER (I) = SPACE
878      087800          GO TO 272-PACK.
879      087900      ADD 1 TO J.
880      088000      MOVE SOURCE-CHARACTER (I) TO PAKDCARD (J).
881      088100      IF PAKD4 NOT = ≠GOTO≠
882      088200          GO TO 272-PACK.
883      088300      MOVE 0 TO J.
884      088400
885      088500  273-MOVE-COND.
886      088600      ADD 1 TO I.
887      088700      IF I IS GREATER THAN K
888      088800          GO TO 274-EXTRACT-ADR.
889      088900      IF SOURCE-CHARACTER (I) = SPACE
890      089000          GO TO 273-MOVE-COND.
891      089100      IF SOURCE-CHARACTER (I) = ≠.≠
892      089200          GO TO 273-MOVE-COND.
893      089300      ADD 1 TO J.
894      089400      MOVE SOURCE-CHARACTER (I) TO GO-TO-COND (J).
895      089500      GO TO 273-MOVE-COND.
896      089600
897      089700  274-EXTRACT-ADR.
898      089800      ADD 1 TO J.
899      089900      MOVE ≠.≠ TO GO-TO-COND (J).
900      090000      ADD 1 TO J.
901      090100      MOVE ≠F≠ TO GO-TO-COND (J).
902      090200      ADD 1 TO J.
903      090300      MOVE ≠0≠ TO GO-TO-COND (J).
904      090400      ADD 1 TO J.
```

```
905      090500      MOVE ≠.≠ TO GO-TO-COND (J).
906      090600      ADD 1 TO K.
907      090700      PERFORM 910-GENERATE-LABEL.
908      090800      PERFORM 979-NEXT-IS-GENERATED THRU 980-EXIT.
909      090900      MOVE GOTO-NEXT TO ELSE-FIELD.
910      091000      MOVE 16 TO ELSE-LENGTH.
911      091100      MOVE SPACE TO GOTO-ADR.
912      091200      MOVE J TO J1.
913      091300      MOVE 0 TO I.
914      091400
915      091500 275-EXTRACT-ADR.
916      091600      ADD 1 TO K.
917      091700      IF SOURCE-CHARACTER (K) NOT NUMERIC
918      091800          GO TO 276-WRITE-EM.
919      091900      ADD 1 TO I.
920      092000      ADD 1 TO J.
921      092100      MOVE SOURCE-CHARACTER (K) TO GO-TO-COND (J).
922      092200      MOVE SOURCE-CHARACTER (K) TO GOTO-DIGT (I).
923      092300      GO TO 275-EXTRACT-ADR.
924      092400
925      092500 276-WRITE-EM.
926      092600      MOVE COMPUTED-GO-TO TO CONDTION.
927      092700      MOVE J TO COND-LENGTH.
928      092800      PERFORM 981-INIT-COND THRU 984-INIT-COND-END.
929      092900      MOVE LADJGOTO TO RESALT.
930      093000      ADD 6 TO I GIVING RES-LENGTH.
931      093100      MOVE SPACE TO GOTO-ADR.
932      093200      MOVE 0 TO I.
933      093300      IF SOURCE-CHARACTER (K) = ≠.≠
934      093400          PERFORM 985-WRITE-GROUP THRU 989-WRITE-GROUP-END
935      093500          PERFORM 910-GENERATE-LABEL
936      093600          PERFORM 979-NEXT-IS-GENERATED THRU 980-EXIT
937      093700          MOVE GOTO-NEXT TO ELSE-FIELD
938      093800          GO TO 275-EXTRACT-ADR.
939      093900      MOVE ≠END≠ TO ELSE-FIELD.
940      094000      MOVE 3 TO ELSE-LENGTH.
941      094100      PERFORM 985-WRITE-GROUP THRU 989-WRITE-GROUP-END.
942      094200      GO TO 290-GO-FINISHED.
943      094300
944      094400 278-COMPUTED-GO-TO.
945      094500      EXAMINE BODY TALLYING UNTIL FIRST ≠)≠.
946      094600      MOVE TALLY TO I.
947      094700      ADD 1 TO I.
948      094800      MOVE 1 TO J.
949      094900      MOVE SPACES TO COMPUTED-GO-TO.
950      095000
951      095100 279-EXTRACT-INDEX.
952      095200      ADD 1 TO I.
953      095300      IF I IS GREATER THAN SOURCE-LENGTH GO TO 280-GENERATE-IFS.
954      095400      IF SOURCE-CHARACTER (I) = ≠.≠ OR SPACES
955      095500          GO TO 279-EXTRACT-INDEX.
956      095600      MOVE SOURCE-CHARACTER (I) TO GO-TO-COND (J).
957      095700      ADD 1 TO J.
958      095800      GO TO 279-EXTRACT-INDEX.
959      095900
960      096000 280-GENERATE-IFS.
```

```
961      096100      MOVE ≠.≠ TO GO-TO-COND (J).
962      096200      ADD 1 TO J.
963      096300      MOVE ≠E≠ TO GO-TO-COND (J).
964      096400      ADD 1 TO J.
965      096500      MOVE ≠0≠ TO GO-TO-COND (J).
966      096600      ADD 1 TO J.
967      096700      MOVE ≠.≠ TO GO-TO-COND (J).
968      096800      ADD 1 TO J.
969      096900      MOVE 1 TO NUM-GO-TO-COND (J).
970      097000      EXAMINE BODY TALLYING UNTIL FIRST ≠(≠.
971      097100      MOVE TALLY TO I.
972      097200      ADD 1 TO I.
973      097300      MOVE 1 TO K.
974      097400      PERFORM 910-GENERATE-LABEL.
975      097500      PERFORM 979-NEXT-IS-GENERATED THRU 980-EXIT.
976      097600      MOVE GOTO-NEXT TO ELSE-FIELD.
977      097700      MOVE 16 TO ELSE-LENGTH.
978      097800      MOVE SPACES TO GOTO-ADR.
979      097900
980      098000  285-GENERATE-IFS.
981      098100      PERFORM 912-INCR-I.
982      098200      IF SOURCE-CHARACTER (I) = SPACES GO TO 285-GENERATE-IFS.
983      098300      IF SOURCE-CHARACTER (I) = ≠.≠
984      098400          PERFORM 951-WRITE-GROUP
985      098500          PERFORM 910-GENERATE-LABEL
986      098600          GO TO 285-GENERATE-IFS.
987      098700      IF SOURCE-CHARACTER (I) = ≠)≠
988      098800          PERFORM 977-FIND-NEXT-LABEL THRU 980-EXIT
989      098900          MOVE GOTO-NEXT TO ELSE-FIELD
990      099000          MOVE 16 TO ELSE-LENGTH
991      099100          PERFORM 951-WRITE-GROUP
992      099200          GO TO 290-GO-FINISHED.
993      099300      MOVE SOURCE-CHARACTER (I) TO GOTO-DIGT (K).
994      099400      ADD 1 TO K.
995      099500      GO TO 285-GENERATE-IFS.
996      099600
997      099700  290-GO-FINISHED.
998      099800      IF CARD-LABL = SPACE
999      099900          PERFORM 910-GENERATE-LABEL.
1000     100000      GO TO 110-READ.
1001     100100
1002     100200  295-CLOSE.
1003     100300      CLOSE INTER1.
1004     100400
1005     100500  300-REOPEN.
1006     100600      MOVE SPACE TO RES-LNTHX.
1007     100700      MOVE SPACE TO ELSE-LNTHX.
1008     100800      OPEN INPUT INTER1.
1009     100900
1010     101000  303-READ.
1011     101100      READ INTER1 AT END
1012     101200          MOVE +0 TO RETN-CODE
1013     101300          GO TO THE-END.
1014     101400
1015     101500  305-CHECK.
1016     101600      IF CHAR1 = ≠N≠
```

```
1017     101700          PERFORM 330-NEW-TABLE THRU 340-EXIT.
1018     101800      IF CHAR1 NOT = ≠-≠ AND NOT = ≠ ≠
1019     101900          MOVE ≠0≠ TO GOTO-FLAG.
1020     102000      IF CHAR1 NOT = ≠R≠ AND NOT = ≠E≠ AND NOT = ≠A≠
1021     102100          GO TO 350-WRITE.
1022     102200      MOVE CHAR1 TO LAST-CHAR.
1023     102300      MOVE 1 TO I.
1024     102400      MOVE 1 TO J.
1025     102500      MOVE SPACE TO PACK-TABLE.
1026     102600
1027     102700 310-PACK.
1028     102800      IF INT-CHARACTER (I) NOT = SPACE
1029     102900          MOVE INT-CHARACTER (I) TO PAKDCARD (J)
1030     103000          ADD 1 TO J.
1031     103100      ADD 1 TO I.
1032     103200      IF I IS GREATER THAN INT-LENGTH
1033     103300          GO TO 320-CHECK-GO.
1034     103400      IF J IS GREATER THAN 8
1035     103500          GO TO 320-CHECK-GO.
1036     103600      GO TO 310-PACK.
1037     103700
1038     103800 320-CHECK-GO.
1039     103900      IF PAKD4 NOT = ≠GOTO≠
1040     104000          GO TO 345-CHECK-CONTINUE.
1041     104100      EXAMINE INSTMNT TALLYING UNTIL FIRST ≠=≠.
1042     104200      IF TALLY IS LESS THAN INT-LENGTH
1043     104300          GO TO 350-WRITE.
1044     104400      MOVE 0 TO CNTR-GP01.
1045     104500      MOVE ≠1≠ TO GOTO-FLAG.
1046     104600      PERFORM 967-COUNT-NON-BLANKS THRU 969-EXIT.
1047     104700      IF CNTR-GP01 IS GREATER THAN 5
1048     104800          GO TO 350-WRITE.
1049     104900      MOVE SPACE TO LABL.
1050     105000      MOVE 1 TO I, J.
1051     105100      PERFORM 970-EXTRACT-ADR THRU 975-EXIT.
1052     105200      IF CHAR1 = ≠R≠
1053     105300          MOVE LADJGOTO TO RESALT
1054     105400          MOVE RESULT-REC TO INTREC2
1055     105500          GO TO 390-WRITE.
1056     105600      IF CHAR1 = ≠F≠
1057     105700          MOVE LADJGOTO TO ELSE-FIELD
1058     105800          MOVE ELSE-REC TO INTREC2
1059     105900          GO TO 390-WRITE.
1060     106000      MOVE ≠A≠ TO GOTO-CHAR.
1061     106100      MOVE GEN-GOTO TO INTREC2.
1062     106200      GO TO 390-WRITE.
1063     106300
1064     106400 330-NEW-TABLE.
1065     106500      IF GOTO-FLAG = ≠1≠
1066     106600          GO TO 340-EXIT.
1067     106700      MOVE INSTMNT TO GOTO-ADR.
1068     106800      IF LAST-CHAR = ≠-≠
1069     106900          MOVE ≠-≠ TO GOTO-CHAR
1070     107000      ELSE
1071     107100          MOVE ≠A≠ TO GOTO-CHAR.
1072     107200      MOVE GEN-GOTO TO INTREC2.
```

```
1073     107300      WRITE INTREC2.
1074     107400
1075     107500 340-EXIT.
1076     107600      EXIT.
1077     107700
1078     107800 345-CHECK-CONTINUE.
1079     107900      IF PAKD8 = ≠CONTINUE≠
1080     108000          GO TO 303-READ.
1081     108100      IF PAKD3 = ≠END≠
1082     108200          MOVE ≠1≠ TO GOTO-FLAG.
1083     108300
1084     108400 350-WRITE.
1085     108500      MOVE SPACE TO INT-LNTHX.
1086     108600      MOVE INTCARD TO INTREC2.
1087     108700
1088     108800 390-WRITE.
1089     108900      WRITE INTREC2.
1090     109000      GO TO 303-READ.
1091     109100
1092     109200 395-SUBROUTINE.
1093     109300      MOVE ≠0≠ TO PROG-ENDF.
1094     109400      OPEN OUTPUT INTER1.
1095     109500      MOVE ≠00000≠ TO LAST-OLDLABL.
1096     109600      ALTER 165-NON-LABELLED TO PROCEED TO 170-FIRST-TIME.
1097     109700      GO TO 110-READ.
1098     109800
1099     109900 THE-END.
1100     110000      CLOSE INTER1.
1101     110100      IF SORC-FLAG NOT = ≠1≠
1102     110200          GO TO 395-SUBROUTINE.
1103     110300      CLOSE SOURCE-DECK.
1104     110400      CLOSE INTER2.
1105     110500      RETURN.
1106     110600
1107     110700 900-COUNT-BRACKETS.
1108     110800      PERFORM 912-INCR-I.
1109     110900      IF SOURCE-CHARACTER (I) = ≠(≠ ADD 1 TO J ELSE IF
1110     111000      SOURCE-CHARACTER (I) = ≠)≠ SUBTRACT 1 FROM J.
1111     111100
1112     111200 905-SQUEEZE-RIGHT.
1113     111300      MOVE 3 TO I.
1114     111400      MOVE 4 TO J.
1115     111500      PERFORM 907-MOVE 3 TIMES.
1116     111600      MOVE FIRST-DIGIT TO LABEL-DIGIT (1).
1117     111700      MOVE 0 TO FIRST-DIGIT.
1118     111800      IF LABEL-DIGIT (4) = SPACES GO TO 905-SQUEEZE-RIGHT.
1119     111900
1120     112000 906-EXIT.
1121     112100      EXIT.
1122     112200
1123     112300 907-MOVE.
1124     112400      MOVE LABEL-DIGIT (I) TO LABEL-DIGIT (J).
1125     112500      SUBTRACT 1 FROM I.
1126     112600      SUBTRACT 1 FROM J.
1127     112700
1128     112800 910-GENERATE-LABEL.
```

```
1129      112900      IF LABL-FLAG = ≠0≠
1130      113000          ADD 1 TO LAST-ADITION
1131      113100          MOVE LAST-LABEL TO LABEL-RECORD
1132      113200          MOVE LAST-ADITION TO EXTRA-PART
1133      113300          MOVE 10 TO INT-LENGTH
1134      113400          MOVE ≠N≠ TO CHAR1
1135      113500          WRITE LABEL-RECORD
1136      113600          MOVE ≠1≠ TO LABL-FLAG.
1137      113700
1138      113800 912-INCR-I.
1139      113900      ADD 1 TO I.
1140      114000      IF I IS GREATER THAN SOURCE-LENGTH
1141      114100          MOVE +2 TO EROR-INDX
1142      114200          MOVE BODY TO EROR-VALU
1143      114300          PERFORM 1040-ERROR-ROUTINE THRU 1049-ERROR-ROUTINE-END
1144      114400          GO TO 110-READ.
1145      114500
1146      114600 915-WRITE-CONDITION.
1147      114700      EXAMINE BODY TALLYING UNTIL FIRST ≠(≠.
1148      114800      MOVE TALLY TO I, K.
1149      114900      ADD 2 TO I.
1150      115000      ADD 1 TO K.
1151      115100      MOVE 1 TO J.
1152      115200      MOVE SPACES TO CONDTION.
1153      115300      IF SOURCE-CHARACTER (I) = ≠(≠
1154      115400          ADD 1 TO J.
1155      115500      PERFORM 920-EXTRACT-CONDITION UNTIL J = 0.
1156      115600      MOVE L TO COND-LENGTH.
1157      115700
1158      115800 920-EXTRACT-CONDITION.
1159      115900      COMPUTE L = I - K.
1160      116000      MOVE SOURCE-CHARACTER (I) TO COND-CHAR (L).
1161      116100      PERFORM 912-INCR-I.
1162      116200      IF SOURCE-CHARACTER (I) = ≠(≠ ADD 1 TO J ELSE IF
1163      116300      SOURCE-CHARACTER (I) = ≠)≠ SUBTRACT 1 FROM J.
1164      116400
1165      116500 930-WRITE-RESULT.
1166      116600      MOVE 1 TO L.
1167      116700      MOVE SPACES TO RESALT.
1168      116800      PERFORM 935-EXTRACT-RESULT UNTIL
1169      116900      I IS GREATER THAN SOURCE-LENGTH.
1170      117000      ADD 3 TO L GIVING RES-LENGTH.
1171      117100      WRITE INTCARD FROM RESULT-REC.
1172      117200      MOVE ≠0≠ TO LABL-FLAG.
1173      117300
1174      117400 935-EXTRACT-RESULT.
1175      117500      MOVE SOURCE-CHARACTER (I) TO RES-CHAR (L).
1176      117600      ADD 1 TO I.
1177      117700      ADD 1 TO L.
1178      117800
1179      117900 940-EXTRACT-ADR.
1180      118000      PERFORM 912-INCR-I.
1181      118100      IF SOURCE-CHARACTER (I) NOT NUMERIC GO TO 941-EXIT.
1182      118200      MOVE SOURCE-CHARACTER (I) TO GOTO-DIGT (J).
1183      118300      ADD 1 TO J.
1184      118400
```

```
1185      118500 941-EXIT.
1186      118600     EXIT.
1187      118700
1188      118800 943-COUNT-COMMAS.
1189      118900     ADD 1 TO J.
1190      119000     IF SOURCE-CHARACTER (J) = ≠,≠ ADD 1 TO L.
1191      119100
1192      119200 945-WRITE-LESSTHAN.
1193      119300     ADD 1 TO L.
1194      119400     ADD 1 TO K.
1195      119500     MOVE LESTHAN (L) TO COND-CHAR (K).
1196      119600
1197      119700 947-WRITE-EQUAL.
1198      119800     ADD 1 TO L.
1199      119900     ADD 1 TO K.
1200      120000     MOVE EQUAL (L) TO COND-CHAR (K).
1201      120100
1202      120200 951-WRITE-GROUP.
1203      120300     MOVE LADJGOTO TO RESALT.
1204      120400     ADD 5 TO K GIVING RES-LENGTH.
1205      120500     MOVE SPACES TO GOTO-ADR.
1206      120600     MOVE 1 TO K.
1207      120700     MOVE COMPUTED-GO-TO TO CONDTION.
1208      120800     MOVE J TO COND-LENGTH.
1209      120900     IF NUM-GO-TO-COND (J) = 9
1210      121000         IF GO-TO-COND (J - 1) = ≠,≠
1211      121100             MOVE 1 TO NUM-GO-TO-COND (J)
1212      121200             ADD 1 TO J
1213      121300             MOVE 0 TO NUM-GO-TO-COND (J)
1214      121400         ELSE ADD 1 TO NUM-GO-TO-COND (J - 1)
1215      121500             MOVE 0 TO NUM-GO-TO-COND (J)
1216      121600     ELSE ADD 1 TO NUM-GO-TO-COND (J).
1217      121700     WRITE INTCARD FROM COND-REC.
1218      121800     WRITE INTCARD FROM RESULT-REC.
1219      121900     WRITE INTCARD FROM ELSE-REC.
1220      122000     MOVE ≠0≠ TO LABL-FLAG.
1221      122100
1222      122200 953-CHECK-DO-STACK.
1223      122300     IF STACK-LENGTH = 0 GO TO 955-EXIT.
1224      122400     IF LABL = DO-END-LABL (STACK-LENGTH) NEXT SENTENCE
1225      122500         ELSE GO TO 955-EXIT.
1226      122600     MOVE DO-TST (STACK-LENGTH) TO CONDTION.
1227      122700     MOVE DO-COND-LENGTH (STACK-LENGTH) TO COND-LENGTH.
1228      122800     PERFORM 910-GENERATE-LABEL.
1229      122900     PERFORM 977-FIND-NEXT-LABEL THRU 980-EXIT.
1230      123000     MOVE GOTO-NEXT TO RESALT.
1231      123100     MOVE 16 TO RES-LENGTH.
1232      123200     MOVE DO-RETN (STACK-LENGTH) TO GOTO-ADR.
1233      123300     MOVE DO-RSLT (STACK-LENGTH) TO ELSE-FIELD.
1234      123400     MOVE DO-INCR-LENGTH (STACK-LENGTH) TO ELSE-LENGTH.
1235      123500     WRITE INTCARD FROM COND-REC.
1236      123600     WRITE INTCARD FROM RESULT-REC.
1237      123700     WRITE INTCARD FROM ELSE-REC.
1238      123800     MOVE LADJGOTO TO ELSE-FIELD.
1239      123900     MOVE 16 TO ELSE-LENGTH.
1240      124000     WRITE INTCARD FROM ELSE-REC.
```

```
1241     124100      MOVE ≠0≠ TO LABL-FLAG.
1242     124200      IF CARD-LABL = SPACE
1243     124300          PERFORM 910-GENERATE-LABEL.
1244     124400      SUBTRACT 1 FROM STACK-LENGTH.
1245     124500
1246     124600 955-EXIT.  EXIT.
1247     124700
1248     124800 957-STORE-SOURCE.
1249     124900      ADD 1 TO SOURCE-CRDS.
1250     125000      MULTIPLY SOURCE-CRDS BY 66 GIVING SOURCE-LENGTH.
1251     125100      IF SOURCE-LENGTH IS GREATER THAN 1320
1252     125200          MOVE +3 TO EROR-INDX
1253     125300          PERFORM 1040-ERROR-ROUTINE THRU 1049-ERROR-ROUTINE-END
1254     125400          MOVE ≠1≠ TO SORC-FLAG
1255     125500          MOVE +99 TO RETN-CODE
1256     125600          GO TO THE-END.
1257     125700      MOVE CARD-BODY TO SOURCE-CRD-STORE (SOURCE-CRDS).
1258     125800
1259     125900 959-EXIT.
1260     126000      EXIT.
1261     126100
1262     126200 961-WRITE-INTCARD.
1263     126300      ADD 1 TO I.
1264     126400      IF DLRS-FLAG = ≠1≠
1265     126500          MOVE 0 TO J
1266     126600          MOVE 0 TO K
1267     126700          MOVE SPACE TO INSTMNT
1268     126800          GO TO 963-DOLLARS.
1269     126900      MOVE SOURCE-CRD-STORE (I) TO INSTMNT.
1270     127000      IF SOURCE-LENGTH IS GREATER THAN 71
1271     127100          MOVE 71 TO INT-LENGTH
1272     127200          SUBTRACT 71 FROM SOURCE-LENGTH
1273     127300      ELSE
1274     127400          MOVE SOURCE-LENGTH TO INT-LENGTH.
1275     127500      WRITE INTCARD.
1276     127600      MOVE ≠0≠ TO LABL-FLAG.
1277     127700      IF I = SOURCE-CRDS
1278     127800          GO TO 965-EXIT.
1279     127900      MOVE ≠-≠ TO CHAR1.
1280     128000      GO TO 961-WRITE-INTCARD.
1281     128100
1282     128200 963-DOLLARS.
1283     128300      ADD 1 TO J.
1284     128400      IF J IS GREATER THAN SOURCE-LENGTH
1285     128500          MOVE SOURCE-LENGTH TO INT-LENGTH
1286     128600          WRITE INTCARD
1287     128700          MOVE ≠0≠ TO LABL-FLAG
1288     128800          GO TO 965-EXIT.
1289     128900      ADD 1 TO K.
1290     129000      MOVE SOURCE-CHARACTER (J) TO INT-CHARACTER (K).
1291     129100      GO TO 963-DOLLARS.
1292     129200
1293     129300 965-EXIT.
1294     129400      EXIT.
1295     129500
1296     129600 967-COUNT-NON-BLANKS.
```

```
1297     129700      IF INT-CHARACTER (I) NOT = SPACE
1298     129800          ADD 1 TO CNTR-GP01.
1299     129900      ADD 1 TO I.
1300     130000      IF I IS GREATER THAN INT-LENGTH
1301     130100          GO TO 969-EXIT.
1302     130200      GO TO 967-COUNT-NON-BLANKS.
1303     130300
1304     130400 969-EXIT.
1305     130500      EXIT.
1306     130600
1307     130700 970-EXTRACT-ADR.
1308     130800      IF INT-CHARACTER (I) IS NUMERIC
1309     130900          MOVE INT-CHARACTER (I) TO DIJIT (J)
1310     131000          ADD 1 TO J.
1311     131100      ADD 1 TO I.
1312     131200      IF I IS GREATER THAN INT-LENGTH OR J IS GREATER THAN 5
1313     131300
1314     131400          MOVE 5 TO J
1315     131500          GO TO 972-SQUEEZE.
1316     131600      GO TO 970-EXTRACT-ADR.
1317     131700
1318     131800 972-SQUEEZE.
1319     131900      IF DIJIT (J) NOT NUMERIC
1320     132000          PERFORM 973-SQUEEZE 4 TIMES
1321     132100          MOVE 0 TO DIJIT (1)
1322     132200          MOVE 5 TO J
1323     132300          GO TO 972-SQUEEZE.
1324     132400      MOVE 1 TO I.
1325     132500      GO TO 974-FIND-LABEL.
1326     132600
1327     132700 973-SQUEEZE.
1328     132800      MOVE DIJIT (J - 1) TO DIJIT (J).
1329     132900      SUBTRACT 1 FROM J.
1330     133000
1331     133100 974-FIND-LABEL.
1332     133200      IF LABL = OLDLABL (I)
1333     133300          MOVE OLDLABL (I) TO GOTO-OLD
1334     133400          MOVE ADITION (I) TO GOTO-NEW.
1335     133500          GO TO 975-EXIT.
1336     133600      ADD 1 TO I.
1337     133700      IF I IS GREATER THAN TABLE-LENGTH
1338     133800          MOVE +4 TO EROR-INDX
1339     133900          MOVE LABL TO EROR-VALU
1340     134000          PERFORM 1040-ERROR-ROUTINE THRU 1049-ERROR-ROUTINE-END
1341     134100          MOVE "1" TO SORC-FLAG
1342     134200          MOVE +99 TO RETN-CODE
1343     134300          GO TO THE-END.
1344     134400      GO TO 974-FIND-LABEL.
1345     134500
1346     134600 975-EXIT.  EXIT.
1347     134700
1348     134800 977-FIND-NEXT-LABEL.
1349     134900      IF CARD-LABL  = SPACE
1350     135000          GO TO 979-NEXT-IS-GENERATED.
1351     135100      MOVE 5 TO I1.
1352     135200      MOVE 4 TO I2.
```

```
1353    135300
1354    135400 978-NEXT-EXISTS.
1355    135500     IF I1 = 0
1356    135600         ADD 100 TO LABL-NUMBR GIVING GOTO-NEWLABL
1357    135700         GO TO 980-EXIT.
1358    135800     IF I2 = 0
1359    135900         MOVE 0 TO GOTO-OLDDIG (I1)
1360    136000         SUBTRACT 1 FROM I1
1361    136100         GO TO 978-NEXT-EXISTS.
1362    136200     MOVE LABEL-CDIGIT (I2) TO GOTO-OLDDIG (I1).
1363    136300     IF LABEL-CDIGIT (I2) NOT = SPACE
1364    136400         SUBTRACT 1 FROM I1.
1365    136500     SUBTRACT 1 FROM I2.
1366    136600     IF I2 = 0
1367    136700         MOVE FIRST-CDIGIT TO GOTO-OLDDIG (I1)
1368    136800         IF FIRST-CDIGIT NOT = SPACE
1369    136900             SUBTRACT 1 FROM I1.
1370    137000     GO TO 978-NEXT-EXISTS.
1371    137100
1372    137200 979-NEXT-IS-GENERATED.
1373    137300     MOVE LAST-OLDLABL TO GOTO-OLDLABL
1374    137400     ADD 1 TO LAST-ADITION GIVING GOTO-NEWLABL.
1375    137500
1376    137600 980-EXIT.
1377    137700     EXIT.
1378    137800
1379    137900 981-INIT-COND.
1380    138000     MOVE SPACE TO GO-TO-COND (J).
1381    138100     SUBTRACT 1 FROM J.
1382    138200     IF J IS GREATER THAN J1
1383    138300         GO TO 981-INIT-COND.
1384    138400
1385    138500 984-INIT-COND-END.
1386    138600     EXIT.
1387    138700
1388    138800 985-WRITE-GROUP.
1389    138900     WRITE INTCARD FROM COND-REC.
1390    139000     WRITE INTCARD FROM RESULT-REC.
1391    139100     WRITE INTCARD FROM ELSE-REC.
1392    139200     MOVE ≠0≠ TO LABL-FLAG.
1393    139300
1394    139400 989-WRITE-GROUP-END.
1395    139500     EXIT.
1396    139600
1397    139700 1005-WRITE-LINE.
1398    139800     IF C-FLAG = SPACE
1399    139900         GO TO 1009-WRITE-LINE-END.
1400    140000     MOVE +0 TO PETN-CODE.
1401    140100     MOVE ≠C≠ TO REPT-CODE.
1402    140200     CALL ≠DETAIL91≠ USING COMN-INFO.
1403    140300
1404    140400 1009-WRITE-LINE-END.
1405    140500     EXIT.
1406    140600
1407    140700 1010-COMMENT.
1408    140800     GO TO 1012-COMMENT.
```

```
1409      140900
1410      141000 1012-COMMENT.
1411      141100     ALTER 1010-COMMENT TO PROCEED TO 1014-COMMENT.
1412      141200     PERFORM 1030-FIRST-LABEL THRU 1035-EXIT.
1413      141300
1414      141400 1014-COMMENT.
1415      141500     ADD 1 TO CMNT-LNTH.
1416      141600     IF CMNT-LNTH IS GREATER THAN 20
1417      141700         MOVE 6 TO EROR-INDX
1418      141800         MOVE CARD TO EROR-VALU
1419      141900         PERFORM 1040-ERROR-ROUTINE THRU 1049-ERROR-ROUTINE-END
1420      142000         GO TO 1014-EXIT.
1421      142100     MOVE CARD TO CMNT-RECD (CMNT-LNTH).
1422      142200
1423      142300 1014-EXIT.
1424      142400     EXIT.
1425      142500
1426      142600 1015-WRITE-NONEXEC.
1427      142700     IF E-FLAG = SPACE
1428      142800         GO TO 1019-END.
1429      142900     ADD 1 TO I.
1430      143000     IF I IS GREATER THAN SOURCE-CRDS
1431      143100         GO TO 1019-END.
1432      143200     MOVE +0 TO RETN-CODE.
1433      143300     MOVE ≠E≠ TO REPT-CODE.
1434      143400     MOVE SOURCE-CRD-STORE (I) TO NONX-INFO.
1435      143500     CALL ≠DETBL91≠ USING COMN-INFO.
1436      143600     GO TO 1015-WRITE-NONEXEC.
1437      143700
1438      143800 1019-END.
1439      143900     EXIT.
1440      144000
1441      144100 1030-FIRST-LABEL.
1442      144200     IF LAST-OLDLABL NOT = ≠**/**≠
1443      144300         GO TO 1033-TEST-SUB.
1444      144400     MOVE ≠00000≠ TO LAST-OLDLABL FIRST-PART.
1445      144500     MOVE LAST-ADITION TO EXTRA-PART.
1446      144600     MOVE ≠N≠ TO CHAR1.
1447      144700     MOVE 10 TO INT-LENGTH.
1448      144800     WRITE LABEL-RECORD.
1449      144900     MOVE ≠1≠ TO LABL-FLAG.
1450      145000     GO TO 1035-EXIT.
1451      145100
1452      145200 1033-TEST-SUB.
1453      145300     IF LAST-OLDLABL NOT = ≠00000≠
1454      145400         GO TO 1035-EXIT.
1455      145500     MOVE ≠0≠ TO LABL-FLAG.
1456      145600     PERFORM 910-GENERATE-LABEL.
1457      145700
1458      145800 1035-EXIT.
1459      145900     EXIT.
1460      146000
1461      146100 1040-ERROR-ROUTINE.
1462      146200     MOVE EROR-MESG (EROR-INDX) TO MESG-AREA.
1463      146300     MOVE EROR-INDX TO MESG-NMBR.
1464      146400     DISPLAY MESG-LINE.
```

```
1465     146500      MOVE SPACE TO EROR-VALU MESG-AREA.
1466     146600
1467     146700 1049-ERROR-ROUTINE-END.
1468     146800      EXIT.
1469     146900
1470     147000 1050-WRITE-COMMENTS.
1471     147100      MOVE +0 TO INDX-GP02.
1472     147200
1473     147300 1052-MOVE-AND-WRITE.
1474     147400      ADD +1 TO INDX-GP02.
1475     147500      IF INDX-GP02 IS GREATER THAN CMNT-LNTH
1476     147600          MOVE SPACE TO CMNT-TABL
1477     147700          MOVE 0 TO CMNT-LNTH
1478     147800          GO TO 1059-WRITE-COMMENTS-END.
1479     147900      MOVE CMNT-RECD (INDX-GP02) TO INTCARD.
1480     148000      MOVE SPACE TO CHAR1.
1481     148100      MOVE 71 TO INT-LENGTH.
1482     148200      WRITE INTCARD.
1483     148300      GO TO 1052-MOVE-AND-WRITE.
1484     148400
1485     148500 1059-WRITE-COMMENTS-END.
1486     148600      EXIT.
1487     148700
1488     148800
0 LENGTH IS  020206
0R SCM USED
```

```
001    000100 IDENTIFICATION DIVISION.
002    000200 PROGRAM-ID. DETBL90.
003    000300 INSTALLATION. SIR GEORGE WILLIAMS UNIVERSITY.
004    000400 DATE-COMPILED. OCTOBER  75/07/18..
005    000500 REMARKS..
006    000600         THIS PROGRAM PRINTS ALL REPORTS OF THE DECISION TABLE
007    000700     GENERATING SYSTEM.  AVAILABLE REPORTS ARE AS FOLLOWS
008    000800
009    000900         C - ORIGINAL SOURCE LIST
010    001000         E- NON-EXECUTABLE STATEMENTS
011    001100         F - TABLE LISTING
012    001200         H - TABLE CROSS REFERENCE
013    001300
014    001400 ENVIRONMENT DIVISION.
015    001500 CONFIGURATION SECTION.
016    001600
017    001700 SOURCE-COMPUTER. 6400.
018    001800 OBJECT-COMPUTER. 6400.
019    001900 INPUT-OUTPUT SECTION.
020    002000
021    002100 FILE-CONTROL..
022    002200
023    002300     SELECT LNEFILE ASSIGN TO LNEFILE.
024    002400     SELECT PRTFILE ASSIGN TO PRTFILE.
025    002500
026    002600 DATA DIVISION.
027    002700 FILE SECTION.
028    002800 FD  LNEFILE
029    002900     LABEL RECORDS ARE OMITTED
030    003000     BLOCK CONTAINS 20 RECORDS
031    003100     RECORD CONTAINS 133 CHARACTERS
032    003200     DATA RECORD IS PRNT-RECD.
033    003300
034    003400 01  PRNT-RECD              PIC X(133).
035    003500
036    003600 FD  PRTFILE
037    003700     LABEL RECORDS ARE OMITTED
038    003800     BLOCK CONTAINS 20 RECORDS
039    003900     RECORD CONTAINS 133 CHARACTERS
040    004000     DATA RECORD IS PRNT-LINE.
041    004100
042    004200 01  PRNT-LINE.
043    004300     05  CARR-CONT          PIC X.
044    004400     05  PRNT-INFO          PIC X(132).
045    004500
046    004600 COMMON-STORAGE SECTION.
047    004700 01  COMN-INFO.
048    004800     05  RETN-CODE          PIC S9(4)   COMP.
049    004900     05  PASD-INFO          PIC X(128).
050    005000
051    005100 WORKING-STORAGE SECTION.
052    005200 77  LINE-CNTR              PIC S9(4)   COMP    VALUE +0.
053    005300 77  PAGE-CNTR              PIC S9(4)   COMP    VALUE +0.
054    005400 77  LAST-REPT              PIC X               VALUE SPACE.
055    005500 77  HOST-REPT              PIC X           VALUE SPACE.
056    005600 77  INDX-GP01              PIC S9(4) COMP  VALUE +0.
```

```
057      005700
058      005800 01  ROST-AREA.
059      005900      05  REPT-ROST          PIC X OCCURS 10 TIMES.
060      006000
061      006100 01  REPT-RECD.
062      006200      05  REPT-CODE          PIC X           VALUE SPACE.
063      006300      05  REPT-INFO.
064      006400          10  REPT-CCTL      PIC X           VALUE SPACE.
065      006500          10  REPT-DATA      PIC X(131)      VALUE SPACE.
066      006600
067      006700 01  RPTC-HED1.
068      006800      05  HDC1-FL05          PIC X(20) VALUE ≠1    ** DECISION TAB≠.
069      006900      05  HDC1-FL10          PIC X(20) VALUE ≠LE GENERATOR RUN ** ≠.
070      007000      05  HDC1-FL15          PIC X(20) VALUE ≠    ** SOURCE LISTIN≠.
071      007100      05  HDC1-FL20          PIC X(11) VALUE ≠G **  PAGE ≠.
072      007200      05  HDC1-PAGE          PIC ZZZ9.
073      007300
074      007400 01  RPTE-HED1.
075      007500      05  HDE1-FL05          PIC X(20) VALUE ≠1    ** DECISION TAB≠.
076      007600      05  HDE1-FL10          PIC X(20) VALUE ≠LE GENERATOR RUN ** ≠.
077      007700      05  HDE1-FL15          PIC X(20) VALUE ≠** NON-EXECUTABLE ST≠.
078      007800      05  HDE1-FL20          PIC X(17) VALUE ≠ATEMENTS ** PAGE ≠.
079      007900      05  HDE1-PAGE          PIC ZZZ9.
080      008000
081      008100 01  RPTF-HED1.
082      008200      05  HDF1-FL05          PIC X(20) VALUE ≠1    ** DECISION TAB≠.
083      008300      05  HDF1-FL10          PIC X(20) VALUE ≠LE GENERATOR RUN ** ≠.
084      008400      05  HDF1-FL15          PIC X(20) VALUE ≠    ** TABLE LISTING ≠.
085      008500      05  HDF1-FL20          PIC X(11) VALUE ≠**    PAGE ≠.
086      008600      05  HDF1-PAGE          PIC ZZZ9.
087      008700
088      008800 01  RPTH-HED1.
089      008900      05  HDH1-FL05          PIC X(20) VALUE ≠1    ** DECISION TAB≠.
090      009000      05  HDH1-FL10          PIC X(20) VALUE ≠LE GENERATOR RUN ** ≠.
091      009100      05  HDH1-FL15          PIC X(20) VALUE ≠  ** CROSS REFERENCE≠.
092      009200      05  HDH1-FL20          PIC X(11) VALUE ≠ **   PAGE ≠.
093      009300      05  HDH1-PAGE          PIC ZZZ9.
094      009400
095      009500 01  RPT9-HED1.
096      009600      05  HD91-FL05          PIC X(20) VALUE ≠1    ** DECISION TAB≠.
097      009700      05  HD91-FL10          PIC X(20) VALUE ≠LE GENERATOR RUN ** ≠.
098      009800      05  HD91-FL15          PIC X(20) VALUE ≠    * UNKNOWN REPOR≠.
099      009900      05  HD91-FL20          PIC X(11) VALUE ≠T **  PAGE ≠.
100      010000      05  HD91-PAGE          PIC ZZZ9.
101      010100
```

```
0102      010200 PROCEDURE DIVISION USING COMN-INFO.
0103      010300 000-PROGRAM-START SECTION 1.
0104      010400 000-START.
0105      010500     OPEN OUTPUT LNEFILE.
0106      010600     MOVE +0 TO RETN-CODE.
0107      010700     GO TO 999-RETURN.
0108      010800
0109      010900 100-WRITE.
0110      011000
0111      011100     ENTRY ≠DETBL91≠ USING COMN-INFO.
0112      011200 DETBL91.
0113      011300     WRITE PRNT-RECD FROM PASD-INFO.
0114      011400     MOVE +0 TO RETN-CODE.
0115      011500     GO TO 999-RETURN.
0116      011600
0117      011700 200-PRINT SECTION 95.
0118      011800
0119      011900     ENTRY ≠DETBL92≠ USING COMN-INFO.
0120      012000 DETBL92.
0121      012100     OPEN OUTPUT PRTFILE.
0122      012200     MOVE +0 TO INDX-GP01.
0123      012300     MOVE PASD-INFO TO ROST-AREA.
0124      012400
0125      012500 210-INITIALIZE-REPTS.
0126      012600     CLOSE LNEFILE.
0127      012700     ADD +1 TO INDX-GP01.
0128      012800     IF INDX-GP01 IS GREATER THAN +10
0129      012900         GO TO 900-TERMINATION.
0130      013000     MOVE REPT-ROST (INDX-GP01) TO ROST-REPT.
0131      013100     IF ROST-REPT = SPACE
0132      013200         GO TO 900-TERMINATION.
0133      013300     OPEN INPUT LNEFILE.
0134      013400
0135      013500 300-READ-INPUT.
0136      013600     READ LNEFILE INTO REPT-RECD AT END
0137      013700         GO TO 210-INITIALIZE-REPTS.
0138      013800     IF REPT-CODE NOT = ROST-REPT
0139      013900
0140      014000         GO TO 300-READ-INPUT.
0141      014100     IF REPT-CODE NOT = LAST-REPT
0142      014200         MOVE +70 TO LINE-CNTR
0143      014300         MOVE +0 TO PAGE-CNTR.
0144      014400     MOVE REPT-CODE TO LAST-REPT.
0145      014500     PERFORM 350-REPORT THRU 359-REPORT-END.
0146      014600     GO TO 300-READ-INPUT.
0147      014700
0148      014800 350-REPORT.
0149      014900     IF REPT-CODE = ≠C≠
0150      015000         PERFORM 400-RPTC THRU 419-RPTC-END
0151      015100     ELSE
0152      015200         IF REPT-CODE = ≠E≠
0153      015300             PERFORM 420-RPTE THRU 439-RPTE-END
0154      015400     ELSE
0155      015500         IF REPT-CODE = ≠F≠
0156      015600             PERFORM 440-RPTF THRU 459-RPTF-END
0157      015700     ELSE
```

```
158      015800          IF REPT-CODE = ≠H≠
159      015900              PERFORM 460-RPTH THRU 479-RPTH-END
160      016000      ELSE
161      016100          PERFORM 880-UNKNOWN THRU 898-UNKNOWN-END.
162      016200
163      016300 359-REPORT-END.
164      016400
165      016500      EXIT.
166      016600
167      016700 400-RPTC.
168      016800      ADD +1 TO LINE-CNTR.
169      016900      IF LINE-CNTR IS GREATER THAN +60
170      017000          PERFORM 410-HEDC THRU 418-HEDC-END.
171      017100      MOVE REPT-INFO TO PRNT-INFO.
172      017200      MOVE SPACE TO CARR-CONT.
173      017300      PERFORM 870-WRITE THRU 879-WRITE-END.
174      017400      GO TO 419-RPTC-END.
175      017500
176      017600 410-HEDC.
177      017700      ADD +1 TO PAGE-CNTR.
178      017800      MOVE PAGE-CNTR TO HDC1-PAGE.
179      017900      MOVE RPTC-HED1 TO PRNT-LINE.
180      018000      PERFORM 870-WRITE THRU 879-WRITE-END.
181      018100      PERFORM 870-WRITE THRU 879-WRITE-END.
182      018200      MOVE +3 TO LINE-CNTR.
183      018300
184      018400 418-HEDC-END.
185      018500
186      018600      EXIT.
187      018700
188      018800 419-RPTC-END.
189      018900
190      019000      EXIT.
191      019100
192      019200 420-RPTE.
193      019300
194      019400      ADD +1 TO LINE-CNTR.
195      019500      IF LINE-CNTR IS GREATER THAN 60
196      019600          PERFORM 430-HEDE THRU 438-HEDE-END.
197      019700      MOVE REPT-INFO TO PRNT-INFO.
198      019800      MOVE SPACE TO CARR-CONT.
199      019900      PERFORM 870-WRITE THRU 879-WRITE-END.
200      020000      GO TO 439-RPTE-END.
201      020100
202      020200 430-HEDE.
203      020300
204      020400      ADD +1 TO PAGE-CNTR.
205      020500      MOVE PAGE-CNTR TO HDE1-PAGE.
206      020600      MOVE RPTE-HED1 TO PRNT-LINE.
207      020700      PERFORM 870-WRITE THRU 879-WRITE-END.
208      020800      PERFORM 870-WRITE THRU 879-WRITE-END.
209      020900      MOVE +3 TO LINE-CNTR.
210      021000
211      021100 438-HEDE-END.
212      021200
213      021300      EXIT.
```

```
021400
021500 439-RPTE-END.
021600
021700     EXIT.
021800
021900 440-RPTF.
022000
022100     IF REPT-CCTL = "1"
022200         PERFORM 450-HEDF THRU 458-HEDF-END
022300         GO TO 459-RPTF-END.
022400     MOVE REPT-INFO TO PRNT-LINE.
022500     PERFORM 870-WRITE THRU 879-WRITE-END.
022600     GO TO 459-RPTF-END.
022700
022800 450-HEDF.
022900
023000     ADD +1 TO PAGE-CNTR.
023100     MOVE PAGE-CNTR TO HDF1-PAGE.
023200     MOVE RPTF-HED1 TO PRNT-LINE.
023300     PERFORM 870-WRITE THRU 879-WRITE-END.
023400
023500 458-HEDF-END.
023600
023700     EXIT.
023800
023900 459-RPTF-END.
024000
024100     EXIT.
024200
024300 460-RPTH.
024400
024500     IF REPT-CCTL = "1"
024600         PERFORM 470-HEDH THRU 478-HEDH-END
024700         GO TO 479-RPTH-END.
024800     MOVE REPT-INFO TO PRNT-LINE.
024900     PERFORM 870-WRITE THRU 879-WRITE-END.
025000     GO TO 479-RPTH-END.
025100
025200 470-HEDH.
025300
025400     ADD +1 TO PAGE-CNTR.
025500     MOVE PAGE-CNTR TO HDH1-PAGE.
025600     MOVE RPTH-HED1 TO PRNT-LINE.
025700     PERFORM 870-WRITE THRU 879-WRITE-END.
025800
025900 478-HEDH-END.
026000
026100     EXIT.
026200
026300 479-RPTH-END.
026400
026500     EXIT.
026600
026700 870-WRITE.
026800
026900     WRITE PRNT-LINE.
```

```
270      027000          MOVE SPACE TO PRNT-LINE.
271      027100
272      027200  879-WRITE-END.
273      027300
274      027400          EXIT.
275      027500
276      027600  880-UNKNOWN.
277      027700
278      027800          ADD +1 TO LINE-CNTR.
279      027900          IF LINE-CNTR IS GREATER THAN +60
280      028000              PERFORM 890-HED9 THRU 897-HED9-END.
281      028100          MOVE REPT-INFO TO PRNT-INFO.
282      028200          MOVE SPACE TO CARR-CONT.
283      028300          PERFORM 870-WRITE THRU 879-WRITE-END.
284      028400          GO TO 898-UNKNOWN-END.
285      028500
286      028600  890-HED9.
287      028700
288      028800          ADD +1 TO PAGE-CNTR.
289      028900          MOVE PAGE-CNTR TO HD91-PAGE.
290      029000          MOVE RPT9-HED1 TO PRNT-LINE.
291      029100          PERFORM 870-WRITE THRU 879-WRITE-END.
292      029200          PERFORM 870-WRITE THRU 879-WRITE-END.
293      029300          MOVE +3 TO LINE-CNTR.
294      029400
295      029500  897-HED9-END.
296      029600
297      029700          EXIT.
298      029800
299      029900  898-UNKNOWN-END.
300      030000
301      030100          EXIT.
302      030200
303      030300  899-WRITE-REPORTS-END.
304      030400
305      030500          EXIT.
306      030600
307      030700  900-TERMINATION SECTION 1.
308      030800
309      030900  901-TERMINATE.
310      031000
311      031100          MOVE +0 TO RETN-CODE.
312      031200          CLOSE PRTFILE.
313      031300
314      031400  999-RETURN.
315      031500
316      031600          RETURN.
317      031700
 LENGTH IS  001334
R SCM USED
```

APPENDIX E

MANUAL REDUCTION OF PROGRAM 6

TABLE 000	
NBLANK=(4H)	
NCBLANK=(1H)	
KNU=ME1=ME2=NT1=NT2=0	
LABEL=10	
DUMVAR=1	
LABEL.EQ.17	YYYYYYYNNN
DUMVAR.LT.0	YYYY-YNN
TEMP(I1).EQ.NCBLANK	YNNNN
I.EQ.7	YYYN
TEMP(57).EQ.NCBLANK.AND.TEMP(58).EQ.NCBLANK	YNN
TEMP(57).LT.J1.AND.TEMP(58).LT.J1	YN
DUMVAR.EQ.0	YN
LABEL.EQ.10	YNN
LABEL.EQ.4	YN
DUMVAR=2	XXXXX
I3=I3+1	X
NT1=KNU	XXX
NTCOF(KNU)=NTCOT(KNU)=NBLANK	XXX
DECODE(7,204,STO(8))NRCO(KNU)	XX
ME2=KNU	X
ME1=KNU	X
DUMVAR=3	X
KNU=KNU+1	X
J=12B	X
DECODE(63,202,STO)IOFCO(KNU),(TEMP(I),I=1,58)	X
I1=I3=0	X
I=J=1	X
DUMVAR=4	X X
I2345=1	X X
J2345=2	X X
K2345=6	X X
STOP	X
END	X
GO TO 10	XXXX XXX
GO TO 12	X
GO TO 13	X

TABLE 10	
READ(60,201)XJOB,ICAR,STO	
EOFCKF(60).EQ.1	YYYN
ME1.EQ.0.AND.ME2.EQ.0	YYN
KNU.EQ.1	YN
I1=I3=1	XX
I2=NTR=0	XX
CALL TRASU(I1,KNU,I3)	X
L=1	X
PRINT 114,ME1,ME2	X
PRINT 115,NT1,NT2	X
STOP	X
END	X
KNU=KNU+1	X
J1=12B	X
DECODE(63,202,STO)IOFCO(KNU),(TEMP(I),I=1,58)	X
I1=0	X
J=1	X X
DECODE(3,205,IOFCO(J))(TEMP(I),I=1,3)	X
I=1	X X
I3=0	X
GO TO 12	X
GO TO 171	X
GO TO 861	X

TABLE 12	
I1=I1+1	
TEMP(I1).GE.J1	YYYYYN
TEMP(I1).EQ.NCBLANK	YNNNN
I.EQ.7	YYYN
TEMP(57).EQ.NCBLANK.AND.TEMP(58).EQ.NCBLANK	YNN
TEMP(57).LT.J1.AND.TEMP(58).LT.J1	YN
I3=I3+1	X
NT1=KNU	XXX
NTCOF(KNU)=NTCOT(KNU)=NBLANK	XXX
DECODE(7,204,STO(8))NRCO(KNU)	XX
ME2=KNU	X
ME1=KNU	X
GO TO 10	XXXX
GO TO 13	X X

TABLE 13	
J.GE.8	YYYYYYYYYYYYN
I3.GT.0.AND.I3.LT.8	YYYYNNNNNNNN
I.EQ.7	YYYNYYYNNNNN
I.GE.7	YYYYN
I.EQ.7	YYYN
TEMP(57).EQ.NCBLANK.AND.TEMP(58).EQ.NCBLANK	YNN YNNYNN
TEMP(57).LT.J1.AND.TEMP(58).LT.J1	YN YN YN
DECODE(13,203,STO(I))NUMF(KNU,I),NUMT(KNU,I)	XXXXX
NT1=KNU	XXX XXX
NTCOF(KNU)=NTCOT(KNU)=NBLANK	XXX XXX
DECODE(13,203,STO(I))NTCOF(KNU),NTCOT(KNU)	XXX
DECODE(7,204,STO(8))NRCO(KNU)	XX XX XX
ME2=KNU	X X X
ME1=KNU	X X
I=I+1	X
I3=0	X
J=1	X
J=J+1	X
GO TO 10	XXXXXXXXXXX
GO TO 12	XX

TABLE 171

TEMP(I).EQ.NCBLANK	YYYYNNNNNNNNNNN
I.GE.3	YYYYYYYYYYYN
I2.EQ.1	YYNNYYYYYYYYYN
IOFCO(J).EQ.IOFCO(J-1)	YYNNNNNNN
(J+1).GT.KNU	YYYNNNNN
IOFCO(J+1).EQ.IOFCO(1)	YYNNN
ME3.NE.0.OR.ME4.NE.0.OR.ME5.NE.0	YNNYNYNN
J.GE.KNU	YNYNYN YN YN
NT2=1	XXXX
IOFCO(J)=IOFCO(J-1)	XX
CALL TRASU(I1,J-1,I3)	XXX XXX
I1=J	XX XX
I1-I3=2	XX
CALL TRASU(1,3,2)	XX
I2=J	X
I=I+1	X
CALL TRASU(I1,KNU,I3)	X X X X X X
L=1	X X X XX XXXX
J=J+1	X X X X X
DECODE(3,205,IOFCO(J))(TEMP(I),I=1,3)	X X X X X
I=1	X X X X X
GO TO 861	X X X XX XXXX
GO TO 171	X X X X X X
GO TO 240	X

TABLE 240	
I2.EQ.1	YYN
J.GE.KNU	YN
CALL TRASU(I1;KNU,I3)	X
L=1	X
J=J+1	X
DECODE(3,205,IOFCO(J))(TEMP(I),I=1,3)	X
I=1	X
I2=I2-1	X
IOFCO(I2)=IOFCO(J)	X
GO TO 861	X
GO TO 171	X
GO TO 240	X

TABLE 861	
J2=L-1	
PRINT 111,J2	
PRINT 112,(NTHD(L,J),J=1,5)	
PRINT 113,(NU2H(L,J),J=1,5)	
L.GE.NTR	YN
PRINT 115,NT1,NT2	X
STOP	X
END	X
L=L+1	X
GO TO 861	X

LIST OF REFERENCES

1. Bohm, Corrado and Jacopini, Giuseppe, Flowdiagrams, Turing Machines and Languages with only Two Formation Rules, Comm. ACM 9 (1966), 366-371.
2. Cavouras, John C. Onthe Conversion of Programs to Decision Tables: Method and Objectives, Comm. ACM 17,8 (August 1974), 456-462.
3. Chapin, N. Flowchart Packages and the ANSI Standard, Datamation (September 1972), 48-53.
4. Dijkstra, E.W. Structured Programming, in Software Engineering Techniques, eds. J.N. Burton and B. Randell (NATO Science Committee, 1969), 88-93.
5. Ganapathy, S. and Rajaraman, V. Information Theory Applied to the Conversion of Decision Tables to Computer Programs, Comm. ACM 16,9 (September 1973), 532-539.
6. Kavanaugh, Thomas F. Tabsol - a Fundamental Concept for Systems Oriented Languages, Proc. Eastern JCC, (1960), 117-127.
7. Kavanaugh, T.F. Decision Structure Tables - a Technique for Business Decision Making, The Journal of Industrial Engineering, (September-October 1963), 249-258.
8. King, P.J.H. and Johnson, R.G. Some Comments on the Use of Ambiguous Decision Tables and Their Conversion

to Computer Programs, Comm. ACM 16,5 (May 1973), 287-290.

9. Muthukrishnan, C.R. and Rajaraman, V. On the Conversion of Decision Tables to Computer Programs, Comm ACM 13,6 (June 1970), 347-351.

10. Schmidt, D.T. and Kavanaugh, T.F. Using Decision Structure Tables, Datamation (March 1964), 48-54.

11. Sherman, Philip M. Flowtrace, A Computer Program for Flowcharting Programs, Comm. ACM 9,12 (December 1966), 845-854.